Low Power Finite Field Multiplication with Wireless Security Applications

Shoaleh Hashemi Namin
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/5638
Low Power Finite Field Multiplication with Wireless Security Applications

By

Shoaleh Hashemi Namin

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2015

© 2015 Shoaleh Hashemi Namin
Low Power Finite Field Multiplication with Wireless Security Applications

by

Shoaleh Hashemi Namin

APPROVED BY:

______________________________________________
M. M. Jamali, External Examiner
Electrical Engineering and Computer Science Department, University of Toledo

______________________________________________
W. Abdul-Kader
Department of Mechanical, Automotive & Materials Engineering, University of Windsor

______________________________________________
K. Tepe
Department of Electrical and Computer Engineering, University of Windsor

______________________________________________
M. Khalid
Department of Electrical and Computer Engineering, University of Windsor

______________________________________________
M. Ahmadi, Co-Advisor
Department of Electrical and Computer Engineering, University of Windsor

______________________________________________
H. Wu, Co-Advisor
Department of Electrical and Computer Engineering, University of Windsor

October 2, 2015
I. DECLARATION OF CO-AUTHORSHIP

I hereby declare that this dissertation incorporates material that is the result of my research conducted under the supervision of my advisors Dr. Majid Ahmadi and Dr. Huapeng Wu. The results of the research are presented in Chapter 5, Chapter 6, and Chapter 7. This dissertation also incorporates the results of my research undertaken in collaboration with Dr. Roberto Muscedere and the results of this research are reported in Chapter 5.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation.

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

II. DECLARATION OF PREVIOUS PUBLICATION

This dissertation includes three original papers that have been previously published/submitted for publication in peer reviewed conferences and journals, as follows:
<table>
<thead>
<tr>
<th>Dissertation Chapter</th>
<th>Publication title/full citation</th>
<th>Publication status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 7</td>
<td>S. Hashemi Namin, H. Wu, and M. Ahmadi, “Low power design for a digit-serial polynomial basis finite field multiplier in $GF(2^m)$,” submitted to IEEE transactions on VLSI systems, 2015.</td>
<td>submitted</td>
</tr>
</tbody>
</table>

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

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

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

Elliptic curve (EC) cryptosystem is the most efficient public key cryptographic system used in practice. However, it is still computationally intensive and has high power consumption, compared to non-public key cryptographic computations. Therefore, for the power constrained wireless devices a power efficient EC cryptosystem is required. Finite field multipliers are main building blocks of an EC cryptosystem. Thus, low power finite field multipliers are required for a power efficient EC cryptosystem.

In this work, we performed VLSI simulation for several existing digit-serial finite field multipliers using the same field with the same VLSI technology so that an effective comparison of their power efficiency along with other parameters can be made. Only few of the digit-serial finite field multiplier architectures proposed in the literature demonstrated power estimation with different VLSI technology for different field sizes. This makes it difficult to compare their power efficiency.

We present a low power design for a digit-serial finite field multiplier in $GF(2^m)$. In the proposed design, factoring technique is used to reduce switching power. Logic gate substitution is also utilized to reduce internal power. Our proposed design along with several existing similar works has been realized on ASIC, and a comparison is made among them. The synthesis results show that the total power consumption is significantly reduced for the proposed multiplier design. Our proposed multiplier design consumes about 27.8% less power than the best existing work in comparison.
DEDICATION

To my parents and sister with love

To my loving husband

To my kind uncle Mike and aunt Mahin
ACKNOWLEDGEMENTS

This work would not be possible without the help and support of many people. At first my sincere gratitude goes to my advisors Dr. Huapeng Wu and Dr. Majid Ahmadi for their precious advice and their financial support throughout my PhD program.

I would like to thank my dissertation committee members, Dr. Mohammed Khalid, and Dr. Kemal Tepe from the Electrical and Computer Engineering Department, Dr. Walid Abdul-Kader from Department of Mechanical, Automotive & Materials Engineering, and Dr. Mohsin M. Jamali from University of Toledo for participating in my seminars, reviewing my dissertation and giving constructive and helpful comments.

Special thanks go to Dr. Roberto Muscedere for his technical support and his guide in preparing the VLSI experiments.

I am also grateful to Andria Ballo, graduate secretary at the Electrical and Computer Engineering Department, for her time and support.

I would like to thank my husband Roohollah Etemadi who has encouraged me and believed in me.

I would like to express my gratitude to my uncle Mike and aunt Mahin for their guidance and constant support. Without their help I would not be able to reach to this point.

Finally, my deepest appreciation goes to my parents and my sister for their unconditional love, encouragement and support.
CONTENTS

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION  ...... iii
ABSTRACT ...................................................................................................................... v
DEDICATION ............................................................................................................. vi
ACKNOWLEDGEMENTS ....................................................................................... vii
LIST OF FIGURES .................................................................................................... xii
LIST OF TABLES ....................................................................................................... xiv
LIST OF ABBREVIATIONS .................................................................................... xv

CHAPTER 1 INTRODUCTION ..................................................................................... 1
1.1 Motivation ........................................................................................................... 1
1.2 Summary of Contributions .............................................................................. 3
1.3 Dissertation Organization .................................................................................. 4

CHAPTER 2 MATHEMATICAL BACKGROUND ...................................................... 7
2.1 Basic Concepts ................................................................................................... 7
2.2 Finite Fields ...................................................................................................... 8
2.3 Binary Extension Fields .................................................................................... 8
2.4 Irreducible Polynomial ...................................................................................... 8
2.5 Representation of Field Elements ...................................................................... 10
2.5.1 Polynomial Basis Representation .............................................................. 10
2.6 Arithmetic in Binary Extension Fields ............................................................. 10
2.6.1 Addition ........................................................................................................ 10
2.6.2 Squaring ....................................................................................................... 10
2.6.3 Inversion ....................................................................................................... 11
2.6.4 Multiplication ............................................................................................... 11
2.7 Example of Polynomial Basis Multiplication .................................................. 12
2.8 Elliptic Curve Cryptography ........................................................................... 12
2.8.1 Elliptic Curves over $GF(2^m)$ ................................................................. 12
2.8.2 Elliptic Curve Point Addition and Doubling ........................................ 13
2.8.3 Point Multiplication .............................................................................. 14
2.8.4 Elliptic curve based cryptographic protocols ........................................ 15
2.9 Conclusion ................................................................................................ 17

CHAPTER 3 ARCHITECTURES FOR DIGIT-SERIAL POLYNOMIAL BASIS
MULTIPLICATION .......................................................................................... 18
3.1 Digit-serial PB Multiplication ................................................................... 18
3.2 Review of Existing Digit-Serial Multipliers ............................................... 19
  3.2.1 Song and Parhi’s Multipliers ............................................................... 20
  3.2.2 Tang et al. Multiplier ........................................................................... 21
  3.2.3 Meher’s Multiplier ............................................................................. 23
  3.2.4 Kumar et al.’s Multiplier .................................................................... 24
3.3 Conclusion ................................................................................................. 24

CHAPTER 4 POWER ESTIMATION IN ASIC AND LOW POWER DESIGN
TECHNIQUES .............................................................................................. 26
4.1 Standard-cell based ASIC design flow ....................................................... 26
  4.1.1 Front-end Design Flow ....................................................................... 27
  4.1.2 Back-end Design Flow ...................................................................... 28
4.2 Power Estimation in ASIC ........................................................................ 28
  4.2.1 Sources of power dissipation in a CMOS standard cell ...................... 28
  4.2.2 Power Consumption Calculation ....................................................... 30
4.3 Low Power Design Techniques ................................................................. 32
  4.3.1 Voltage Scaling Techniques ................................................................. 33
  4.3.2 Switching Minimization Techniques .................................................. 34
  4.3.3 Switched Capacitance Minimization Technique ............................... 36
4.4 Conclusion ................................................................................................. 37
7.2.1 Experimental Settings ............................................................................................. 61
7.2.2 Synthesis Results .................................................................................................. 62
7.3 The effect of digit size on the hardware parameters of the digit-serial PB multipliers ...... 64
  7.3.1 The effect of digit size on area complexity ............................................................... 64
  7.3.2 The effect of digit size on power consumption ....................................................... 65
  7.3.3 The effect of digit size on energy consumption ....................................................... 67
7.4 Conclusion .................................................................................................................. 68

CHAPTER 8 CONCLUSION ......................................................................................... 70
  8.1 Summary and Conclusion ......................................................................................... 70
  8.2 Future Works ........................................................................................................... 72

BIBLIOGRAPHY .......................................................................................................... 74

APPENDIX A COPYRIGHT PERMISSION ............................................................... 78

VITA AUCTORIS ........................................................................................................ 80
LIST OF FIGURES

2.1 ECDH key exchange protocol [1].................................................................16
2.2 The design flow of an EC cryptosystem over binary extension fields...........17
3.1 LSD first multiplier proposed in [14]............................................................21
3.2 MSD first multiplier proposed in [14]............................................................21
3.3 MSD first multiplier proposed in [13] for m=233 and d=8 .........................22
3.4 LSD first multiplier proposed in [15]............................................................24
4.1 Standard-cell based ASIC design flow .......................................................27
4.2 A 2-input NAND cell drives a 2-input NOR cell........................................29
4.3 Realization of function f using (a) linear array structure (b) tree structure.....35
4.4 Factoring technique for switching activity reduction (a) implementation of
the circuit (b) implementation of the circuit with reduced switching activity
[51].............................................................................................................36
4.5 Using gates with identical switching activity and smaller sizes to reduce dynamic
power (a) implementation of $f_{(a)}$ (b) implementation of $f_{(b)}$......................37
5.1 Gate level power estimation flow which is integrated into the synthesis,
gate level simulation, and verification flows...................................................41
5.2 Automated power estimation and optimization flow with initial synthesis
and two levels of synthesis and simulation runs.............................................43
7.1 Proposed digit-serial PB multiplier in $GF(2^m)$........................................53
7.2 $k \times m$ multiplier: (a) without applying factoring; (b) with applied factoring
method (outputs of the shaded modules indicate nets with high switching
activity)...........................................................................................................55
7.3 XOR network 1: (a) before reducing gate count (b) with reduced gate
count ..............................................................................................................57
7.4 (a) Standard design: implementing equation (2) using 24 transistors (b)
Modified design: implementing equation (3) using 20 transistors...............59
7.5 $k \times m$ multiplier with applied factoring, logic gate substitution and gate count
reduction.........................................................................................................59
7.6 Energy per multiplication vs. area...............................................................63
7.7 Area complexity of the multipliers (a) in 65nm technology (b) in 90nm technology
........................................................................................................65

7.8 Total power consumption of the multipliers in (a) 65nm technology (b) 90nm technology
........................................................................................................66

7.9 Energy per multiplication of the multipliers in (a) 65nm technology (b) 90nm technology
........................................................................................................67
LIST OF TABLES

1.1 Key sizes (in bits) for public key cryptography methods...................................2
2.1 Irreducible polynomials of degree from 2 to 12...............................................9
2.2 Irreducible Polynomials......................................................................................9
2.3 Complexity of point operations for the non-supersingular \( E/\text{GF}(2^m) \): \( y^2 + xy = x^3 + ax^2 + b \) in terms of required finite field operations (Multiplication (Mult.), Squaring (Sq.), and Inversion (Inv.)) [35]................................................14
2.4 Fields and irreducible polynomials recommended by NIST and their security lifetime..........................................................15
5.1 Hardware parameters of finite field multipliers in 65nm [52].........................44
5.2 Hardware parameters of finite field multipliers in 90nm [52].........................45
6.1 Space complexities and comparisons...............................................................48
6.2 Time complexities and comparisons...............................................................49
6.3 Power consumption [56]....................................................................................50
6.4 Area, critical path delay, energy consumption and energy-delay product [56]...............................................................................50
7.1 Switching power reduction with factoring method for \( k \times m \) multiplier (experiment setting: \( m = 233, \ k = 8 \))........................................56
7.2 Internal power reduction with logic gate substitution for \( k \times m \) multiplier (experiment setting: \( m = 233, \ k = 8 \))........................................60
7.3 Architecture level area complexities and comparisons................................61
7.4 Time complexities and comparisons...............................................................61
7.5 Power consumption at 50 MHz clock frequency............................................62
7.6 Area, energy consumption, and energy-area product.........................................62
7.7 Area complexity of multipliers in (\( \mu m^2 \)) for various digit sizes in 65nm technology..........................................................64
7.8 Area complexity of multipliers in (\( \mu m^2 \)) for various digit sizes in 90nm technology..........................................................64
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOP</td>
<td>All One Polynomial</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>DAM</td>
<td>Double Accumulator Multiplier</td>
</tr>
<tr>
<td>DB</td>
<td>Dual Basis</td>
</tr>
<tr>
<td>DC</td>
<td>Design Compiler</td>
</tr>
<tr>
<td>ECC</td>
<td>Elliptic Curve Cryptography</td>
</tr>
<tr>
<td>ECDH</td>
<td>Elliptic Curve Diffie–Hellman</td>
</tr>
<tr>
<td>ECDLP</td>
<td>Elliptic Curve Discrete Logarithm Problem</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Elliptic Curve Digital Signature Algorithm</td>
</tr>
<tr>
<td>ECIES</td>
<td>Elliptic Curve Integrated Encryption Scheme</td>
</tr>
<tr>
<td>EDA</td>
<td>Electronic Design Automation</td>
</tr>
<tr>
<td>ESP</td>
<td>Equally Spaced Polynomial</td>
</tr>
<tr>
<td>FF</td>
<td>Flip-flop</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>LSD</td>
<td>Least Significant Digit</td>
</tr>
<tr>
<td>MSD</td>
<td>Most Significant Digit</td>
</tr>
<tr>
<td>MTCMOS</td>
<td>Multi-Threshold CMOS</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>NAM</td>
<td>N-Accumulator Multiplier</td>
</tr>
<tr>
<td>NB</td>
<td>Normal Basis</td>
</tr>
<tr>
<td>NEM</td>
<td>nanoelectromechanical</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NMOS</td>
<td>N-type Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>PB</td>
<td>Polynomial Basis</td>
</tr>
<tr>
<td>PC</td>
<td>Power Compiler</td>
</tr>
<tr>
<td>PMOS</td>
<td>P-type Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Indentification</td>
</tr>
<tr>
<td>RSA</td>
<td>Rivest, Shamir, Adleman</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SAIF</td>
<td>Switching Activity Interchange Format</td>
</tr>
<tr>
<td>SDF</td>
<td>Standard Delay Format</td>
</tr>
<tr>
<td>STM</td>
<td>STMicroelectronics</td>
</tr>
<tr>
<td>Tcl</td>
<td>Tool command language</td>
</tr>
<tr>
<td>TSMC</td>
<td>Taiwan Semiconductor Manufacturing Company</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Motivation
Wireless communication has become ubiquitous due to the growing usage of portable devices such as laptops, tablets, cell phones, smart cards, RFID tags, and sensors. We use wireless devices almost every day for activities such as banking transactions, online shopping, video conferencing and exchanging business information.

The information exchanged through wireless networks is crucial and it must be protected. Therefore, security services such as integrity, confidentiality, authentication and non-repudiation in such networks are essential to protect the information. These services are provided in wireless networks using cryptographic mechanisms. Public key cryptography is a popular method that is used in digital signature and key exchange algorithms designed for wireless communication environment to provide authentication, and key management services.

There are three public key cryptography techniques, namely RSA, ElGamal, and Elliptic Curve Cryptography (ECC). Since ECC uses shorter key size compared to RSA and ElGamal to provide the same level of security, ECC is probably more widely used technique in resource-constrained wireless devices. Table 1.1 shows key size comparisons of ECC with two other public key cryptography methods [1]. Each column presents key sizes that provide the same level of security for different public key cryptography methods. For instance, to achieve the same level of security RSA and
ElGamal require 1024-bit key length while ECC needs only key size of 160 bits. However, ECC is computationally expensive. Therefore, ECC systems require high power to meet the computational requirements.

<table>
<thead>
<tr>
<th>Public key cryptography methods</th>
<th>Key size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>1024</td>
</tr>
<tr>
<td>ElGamal</td>
<td>1024</td>
</tr>
<tr>
<td>ECC</td>
<td>160</td>
</tr>
</tbody>
</table>

Wireless and portable devices have limited sources of power. In battery-operated wireless and mobile devices, small size and low weight are desired. For this reason smaller batteries are used which have limited capacity. Current battery technology cannot meet the power requirements imposed by the applications of the portable wireless devices [2]. That is why, power and energy consumption are important constraints for battery-powered wireless devices. Conservation of power becomes more critical when security algorithms are executed on such devices [3]. One method to save power is to conserve the power spent by the hardware used in computationally expensive security applications such as ECC.

In this dissertation our goal is to reduce the power consumption of an ECC system. Because ECC operations are based on finite field arithmetic, low power design of finite field arithmetic units results in less power consumption which makes the system more suitable for wireless applications.

ECC was introduced in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) in which elliptic curve groups over finite fields are used [4]. Two types of finite fields have been recommended by standards organizations [5]-[8]: prime field, \(GF(p)\), and binary extension field, \(GF(2^m)\). Binary extension field is very attractive for hardware implementation because it offers carry-free arithmetic.

Binary extension field arithmetic includes addition, inversion and multiplication. Addition is realized by bit-wise XOR operation between two field elements. Inversion can be carried out by consecutive field multiplications. Hence, multiplication is the most frequent and important operation among finite field arithmetic operations. The
complexity of these operations is greatly influenced by the representation of the field elements.

There are various methods to represent field elements in $GF(2^m)$, such as polynomial basis (PB), normal basis and dual basis. Organizations which set the standards for cryptography applications have suggested PB as one of the preferred basis [6], [8]. For this reason, a large number of architectures for efficient implementation of PB finite field multipliers have been proposed. These architectures include bit-serial [9], [10], bit-parallel [11], [12], and digit-serial architectures.

A bit-parallel PB multiplier over $GF(2^m)$ accepts two $m$-bit operands and computes the product in one clock cycle. Thus, bit-parallel structure is fast but it is expensive in terms of area. In EC cryptography the binary extension field size, $m$, is required to be in the order of $10^2$. Hence, a bit-parallel structure requires a very high I/O bandwidth which is usually not available in the small portable wireless devices. A bit-serial PB multiplier over $GF(2^m)$ accepts all bits of one operand and the other operand is processed bit by bit. Thus, one multiplication takes $m$ clock cycles making the operation extremely slow for most applications. Nevertheless, bit-serial architecture is area efficient. Digit-serial architecture trades off space for speed; therefore, it achieves moderate speed and area so it is most appropriate for practical use. Many digit-serial PB multipliers have been proposed in the literature [10], [13]–[23]. The main focus of this dissertation is on the digit-serial PB multipliers.

Several architectures have been proposed in the literature, in which low power design techniques have been applied to the bit-serial or bit-parallel [24]–[30] PB finite field multipliers. However, most of the works regarding digit-serial PB multipliers consider optimizing time and area complexities, while much less effort has been done on power optimization [14] or comparing power consumption [18],[31] for these multipliers. Therefore, the main objective of our research is to estimate, compare, and optimize the power and energy consumption of the digit-serial PB multipliers.

1.2 Summary of Contributions
Several existing digit-serial PB multipliers have been implemented using the same VLSI technology, field size, digit size, and power estimation method so that they can be
compared effectively in terms of power and energy consumption, and also area and throughput. Only few works that reported power or energy consumption [13], [14], have used different VLSI technologies, different field sizes and distinct power estimation methodologies which makes it difficult to compare the field multipliers in terms of the power and energy efficiency.

A low power circuit design for a digit-serial PB multiplier in $GF(2^m)$ has been proposed. A factoring based design has been used to reduce switching power. We have also utilized logic gate substitution to reduce internal power. Our proposed design along with several existing similar works has been realized on ASIC, and a comparison is made among them. The synthesis results show that the total power consumption is significantly reduced for the proposed multiplier design. The proposed architecture also consumes the least amount of energy for one multiplication operation among similar existing works.

Moreover, our proposed multiplier along with several existing digit-serial PB multipliers has been implemented with standard digit sizes of 8, 16, 32, and 64 and the effect of digit sizes on their power and energy efficiencies has been studied. The synthesis results show that our proposed multiplier consumes less power and lower energy compared to similar existing multipliers at all of the digit sizes.

We have proposed an automated power analysis and optimization flow. In the commercial semiconductor industry, the synthesis, verification, power analysis, and optimization tasks are automated by digital designers because many parameters need to be tested in a short period of time with no errors. However, in academia there is a lack of skills among most researchers to automate the tasks which can be carried out by electronic design automation (EDA) tools. The proposed automated flow helps the researchers to estimate and optimize the power consumption of designs accurately and effectively with less human efforts while enabling comprehensive design space exploration.

1.3 Dissertation Organization
This dissertation is organized as follows. Chapter 2 briefly introduces mathematical background of finite fields. Binary extension fields and polynomial basis representation of field elements and arithmetic in binary extension fields are provided. Elliptic curve
cryptosystems over binary extension fields, elliptic curve point operations and an elliptic curve cryptography protocol are also discussed in this chapter.

In Chapter 3 digit-serial multiplication scheme is discussed and existing digit-serial PB multipliers are reviewed.

Chapter 4 introduces standard-cell based ASIC design flow which is used for implementation of digit-serial PB multipliers. Sources of power dissipation in standard-cell based CMOS circuits and power calculation in the standard-cell based ASIC design flow are also discussed in this chapter. Finally, several power reduction techniques are described.

In Chapter 5 power estimation and optimization methodology in the standard-cell based ASIC design flow that is used in this research is explained. Automation of the power estimation, design synthesis, and verification is also presented in this chapter. It is shown that using the automated flow a large design space exploration can be performed in a shorter amount of time.

In Chapter 6 polynomial basis multiplication over $GF(2^{283})$ is discussed and a comparative study considering power and energy consumption of some of the existing digit-serial PB multipliers over $GF(2^{283})$ is presented. It is difficult to compare the digit-serial PB multipliers in terms of power or energy consumption because they were implemented using different VLSI technologies and different field sizes and they used different power estimation methods. Therefore, to perform an effective comparison of their power consumption along with other features such as area and critical path delay they have been implemented in the same field, $GF(2^{283})$, with the same digit size, 8, and with the same 0.18µm CMOS technology, and the same method has been used for estimating their power consumption. Synthesis results are analyzed and compared, and recommendations for the most efficient digit-serial PB multiplier are provided for different application constraints.

In Chapter 7, a new low power digit-serial PB multiplier in $GF(2^m)$ is presented. The proposed multiplier and similar multipliers in comparison have been implemented. Synthesis results show that the proposed multiplier consumes lower power and lower energy, and has lower Energy-Area product compared to the similar existing multipliers. The effect of digit size on hardware parameters of the multipliers has also been
examined. The proposed multiplier and similar multipliers in comparison have been implemented using 65nm and 90nm CMOS standard-cell libraries at various digit sizes. Synthesis results show that the proposed multiplier has the lowest area complexity and consumes lower power and lower energy, compared to the similar existing multipliers for all of the various digit sizes. Finally, Chapter 8 presents concluding remarks and future work.
CHAPTER 2
MATHEMATICAL BACKGROUND

This chapter briefly presents mathematical background about finite fields and binary extension fields. This chapter also introduces elliptic curve cryptosystems over binary extension fields, elliptic curve point operations and an elliptic curve cryptography protocol.

2.1 Basic Concepts
Understanding the following algebraic systems is required before introducing finite fields.

Definition 2.1: Let $G$ be a set of elements and $\ast$ a binary operation on $G$, a group $(G,\ast)$ is an algebraic system such that the following properties hold [32]:

- Associativity; that is, for any $x,y,z \in G$, $(x \ast y) \ast z = x \ast (y \ast z)$.
- Existence of an identity element; Let $e \in G$ be the identity element then, for all $x \in G$, $x \ast e = e \ast x = x$.
- Existence of an inverse element for each element in $G$; Let $x^{-1} \in G$ be the inverse element of a given element $x \in G$ then, $x \ast x^{-1} = x^{-1} \ast x = e$.

A group is called an abelian (or a commutative) group if it also satisfies the following property [32]:
- Commutativity; that is, for all $x,y \in G$, $x \ast y = y \ast x$.

Definition 2.2: Let $R$ be a set of elements and $(+,\cdot)$ two binary operations on $R$, a ring $(R,+,\cdot)$ is an algebraic system such that the following properties hold [32]:
- $(R,\ast)$ is an abelian group.
o Associativity with respect to the binary operation (·): that is, for all \(x, y, z \in R\), \((x \cdot y) \cdot z = x \cdot (y \cdot z)\).

o Existence of the distributive property: that is, for all \(x, y, z \in R\),
\[
x \cdot (y + z) = x \cdot y + x \cdot z\]
and \((y + z) \cdot x = y \cdot x + z \cdot x\).

**Definition 2.3:** Let \(F\) be a set of elements and \((+,\cdot)\) two binary operations on \(F\), a field \((F, +, \cdot)\) is an algebraic system such that the following properties hold [32]:

- \((F, +, \cdot)\) is a ring.
- The binary operation \(\cdot\) is commutative.
- Nonzero elements of \(F\) with the binary operation \(\cdot\) form a group.

### 2.2 Finite Fields

A finite field is a field which has finite many elements. A finite field also called Galois field (named after the famous mathematician, Évariste Galois) is denoted by \(GF(q)\) or \(F_q\) where \(q\) represents the number of elements in the finite field and it is called the order of the finite field. The order of a finite field, \(q\), is a prime \((q = p)\) or a power of a prime \((q = p^m)\), where \(m\) is an integer greater than 1. If \((q = p)\) the field is called prime field and denoted by \(GF(p)\). If \((q = p^m)\) the field is called extension field and denoted by \(GF(p^m)\).

### 2.3 Binary Extension Fields

A binary extension field is a type of extension field, \(GF(p^m)\), with \((p = 2)\) and it is denoted by \(GF(2^m)\). A binary extension field \(GF(2^m)\) is generated by an irreducible polynomial \(P(x) = x^m + p_m x^{m-1} + \cdots + p_1 x + 1\) with \(p_i \in GF(2)\) for \(1 \leq i \leq m - 1\) which is explained next. The binary extension field has \(2^m\) elements and various methods are used to represent elements of \(GF(2^m)\).

### 2.4 Irreducible Polynomial

An irreducible polynomial of degree \(m\) is used to generate \(GF(2^m)\). An irreducible polynomial is a polynomial that cannot be factored into two polynomials of smaller degrees. Table 2.1 shows irreducible polynomials of degree from 2 to 12 [33]. For each degree, only one irreducible polynomial is presented as an example.
Table 2.1: Irreducible polynomials of degree from 2 to 12.

<table>
<thead>
<tr>
<th>Irreducible polynomials</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(x) = x^2 + x + 1$</td>
</tr>
<tr>
<td>$p(x) = x^3 + x + 1$</td>
</tr>
<tr>
<td>$p(x) = x^4 + x + 1$</td>
</tr>
<tr>
<td>$p(x) = x^5 + x^2 + 1$</td>
</tr>
<tr>
<td>$P(x) = x^6 + x + 1$</td>
</tr>
<tr>
<td>$P(x) = x^7 + x + 1$</td>
</tr>
<tr>
<td>$P(x) = x^8 + x^4 + x^3 + x + 1$</td>
</tr>
<tr>
<td>$P(x) = x^9 + x + 1$</td>
</tr>
<tr>
<td>$P(x) = x^{10} + x^3 + 1$</td>
</tr>
<tr>
<td>$P(x) = x^{11} + x^2 + 1$</td>
</tr>
<tr>
<td>$P(x) = x^{12} + x^3 + 1$</td>
</tr>
</tbody>
</table>

Four major kinds of irreducible polynomials that are used in the literature for implementation of finite field arithmetic include trinomials, pentanomials, equally-spaced polynomials (ESP) and all one polynomials (AOP). Table 2.2 presents the general expressions of the above mentioned polynomials. Among above mentioned irreducible polynomials, trinomials and pentanomials are more attractive for practical use because they have lower number of non-zero coefficients which results in low complexity architectures. National Institute of Standards and Technology (NIST) recommends irreducible trinomials for the recommended field sizes, if no trinomial exists it recommends pentanomials (see Table 2.4). Irreducible trinomials exist for only 468 values of $m$ smaller than 1024 (≈45.8%). There exists at least one irreducible pentanomial for values of $m$ greater than three [34].

Table 2.2: Irreducible Polynomials

<table>
<thead>
<tr>
<th>Type</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trinomial</td>
<td>$P(x) = x^m + x^k + 1$</td>
</tr>
<tr>
<td>Pentanomial</td>
<td>$P(x) = x^m + x^{k_1} + x^{k_2} + x^{k_3} + 1$</td>
</tr>
<tr>
<td>ESP</td>
<td>$P(x) = x^{ns} + x^{(n-1)s} + \cdots + x^s + 1$, $m = ns$, $1 \leq s \leq m/2$</td>
</tr>
<tr>
<td>AOP</td>
<td>$P(x) = x^m + x^{m-1} + \cdots + x + 1$</td>
</tr>
</tbody>
</table>
2.5 Representation of Field Elements

Elements of a binary extension field, $GF(2^m)$, can be represented in different ways which is called basis. Representation basis has a significant effect on the efficiency of the finite field arithmetic unit. Some of the bases reported in the literature include, polynomial basis (PB), normal basis (NB), dual basis (DB), redundant basis, and shifted polynomial basis. Among these bases, PB and NB are the most popular bases recommended by standard organizations for practical use. Since this dissertation is concentrated on the polynomial basis we introduce the polynomial basis.

2.5.1 Polynomial Basis Representation

Assume that an irreducible polynomial $P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_1x + 1$ with $p_i \in GF(2)$ for $1 \leq i \leq m-1$ generates the finite field. Let $x$ be a root of $P(x)$ then \{1, $x, x^2, \ldots, x^{m-1}\}$ is the polynomial basis and an element $A \in GF(2^m)$ can be represented as a polynomial of degree at most $(m - 1)$ by equation (2.1) or as a binary vector by expression (2.2).

$$A(x) = \sum_{i=0}^{m-1} a_i x^i, \ a_i \in \{0, 1\} \text{ for } 0 \leq i \leq m - 1$$  \hspace{1cm} (2.1)

$$A = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$$  \hspace{1cm} (2.2)

2.6 Arithmetic in Binary Extension Fields

Major operations in binary extension fields are addition, squaring, multiplication, and inversion. These operations are explained in the following sections where we assume polynomial basis for representing field elements.

2.6.1 Addition

Addition of two elements $A(x)$ and $B(x)$ in $GF(2^m)$ can be implemented by bitwise Exclusive-OR (XOR) operation between their corresponding coefficients as shown in equation (2.3). Addition operation in $GF(2^m)$ is a simple and carry-free operation.

$$A(x) + B(x) = (a_{m-1} \oplus b_{m-1})x^{m-1} + (a_{m-2} \oplus b_{m-2})x^{m-2} + \cdots + (a_1 \oplus b_1)x + a_0 \oplus b_0$$  \hspace{1cm} (2.3)

2.6.2 Squaring

Let $A$ be an element in $GF(2^m)$ and assume that an irreducible polynomial, $P(x)$, generates $GF(2^m)$. The square of $A = A(x)$ is obtained by $A^2(x) \mod P(x)$. The square
of $A(x)$ without preforming \textit{“mod $P(x)$”} is given by $A^2(x) = a_{m-1}x^{2m-2} + a_{m-2}x^{2m-4} + \cdots + a_1x^2 + a_0$ that can also be represented by a binary vector $(a_{m-1}, 0, a_{m-2}, 0, a_{m-3}, 0, \ldots, a_1, 0, a_0)$. As can be seen $A^2(x)$ is a polynomial of degree $2m - 2$ that can be obtained from $A(x)$ by adding zero values in between two adjacent coefficients of $A(x)$. In order for the squaring result to be a field element of degree at most $(m - 1)$, the degree of $A^2(x)$ has to be reduced by \textit{“mod $P(x)$”} operation. Squaring can be viewed as a special case of multiplication when two operands are the same; hence, squaring architecture can be optimized and it has lower complexity compared to multiplication module.

\subsection*{2.6.3 Inversion}
Consider an element $A(x)$ in $GF(2^m)$ and an irreducible polynomial, $P(x)$, that generates the finite field. Multiplicative inverse of $A(x)$ is an element of $GF(2^m)$ denoted by $A^{-1}(x)$ and it satisfies the equation $A(x)A^{-1}(x) \equiv 1\, (mod\, P(x))$. The multiplicative inverse of element $A(x)$ exists if $A(x)$ and $P(x)$ are coprime. This condition is always satisfied because $P(x)$ is irreducible. Inversion is the operation that gives the multiplicative inverse of a field element. Inversion can be performed by field multiplications.

\subsection*{2.6.4 Multiplication}
Multiplication in $GF(2^m)$ can be presented by equation (2.4). $A(x), B(x)$ and $C(x)$ are field elements and $P(x)$ is the irreducible polynomial that generates the field. Multiplication of two polynomials of degree at most $(m - 1)$ each, results in a polynomial of degree at most $(2m - 2)$. The degree of this polynomial has to be reduced by \textit{“mod $P(x)$”} operation in order for the product $C(x)$ to be a field element.

\begin{equation}
C(x) = A(x)B(x) \, \text{mod} \, P(x) \quad (2.4)
\end{equation}

Finite field multiplication is a major and complex operation which is used as one of the fundamental arithmetic operations for performing the elliptic curve point multiplication which is explained later in this chapter. Therefore, a large number of algorithms and architectures have been proposed in the literature for efficient implementation of the polynomial basis multipliers.
2.7 Example of Polynomial Basis Multiplication

Consider the binary extension field, $GF(2^7)$ and the irreducible polynomial $P(x) = x^7 + x + 1$ that generates the field. Let $x$ be the root of the irreducible polynomial, $P(x)$, over $GF(2^7)$. Then \{$x^6, x^5, x^4, x^3, x^2, x, 1$\} presents the polynomial basis and each field element $A$ can be represented as follows using this basis:

$$A(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, \ a_i \in \{0,1\}$$

Let $A(x) = x^6 + x + 1$ and $B(x) = x^2$ be two elements of $GF(2^7)$ then $A(x)B(x) \ mod \ p(x)$ can be calculated as follows:

$$A(x)B(x) \ mod \ P(x) = (x^6 + x + 1) \times (x^2) \ mod \ (x^7 + x + 1)$$

$$= (x^8 + x^3 + x^2) \ mod \ (x^7 + x + 1)$$

$$= x^3 + x$$

The result can also be obtained as follows:

Since $x$ is the root of $P(x)$, $x^7 + x + 1 = 0$ or $x^7 = x + 1$. Reducing the degree of $A(x)B(x) = (x^8 + x^3 + x^2)$ can be performed as follows:

$$x^7x + x^3 + x^2 = (x + 1)x + x^3 + x^2$$

$$= x^2 + x + x^3 + x^2$$

$$= x^3 + x$$

2.8 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is a popular public key (also called asymmetric key) method that is based on the elliptic curve discrete logarithm problem which is explained later. ECC uses shorter key compared to RSA and ElGamal public key methods (for instance 160-bit key vs. 1024-bit key) to provide the same level of security. Elliptic curves that are used in cryptography are defined over finite fields. In the following sections elliptic curves defined over $GF(2^m)$ are introduced. Operations that can be performed on the elliptic curve points are discussed. Major operation involved in EC security algorithms is also described. An example of EC cryptosystem is given and the design methodology of an EC cryptosystem is provided.

2.8.1 Elliptic Curves over $GF(2^m)$

There are two types of elliptic curves that are defined over $GF(2^m)$ namely, non-supersingular and supersingular. Supersingular elliptic curves are not suggested for
implementing EC cryptostems because evidences show that EC cryptosystems are less secure when supersingular elliptic curves are used [35]; thus, in the rest of this chapter we only refer to non-supersingular elliptic curves. Equation (2.5) shows non-supersingular elliptic curves, where \( E/\text{GF}(2^m) \) represents an elliptic curve over \( \text{GF}(2^m) \), \( a \in \text{GF}(2^m) \), \( b \in \text{GF}(2^m) \), and \( b \neq 0 \) [36].

\[
\text{non-supersingular } E/\text{GF}(2^m): y^2 + xy = x^3 + ax^2 + b \quad (2.5)
\]

The points on the \( E/\text{GF}(2^m) \) along with point addition operation form an abelian group and \( \infty \) is the identity for addition operation [36]. Group law for the non-supersingular \( E/\text{GF}(2^m) \) is as follows [36]:

- Identity: For all \( P \in E/\text{GF}(2^m) \), \( P + \infty = \infty + P \).
- Negatives: If \( P = (x,y) \in E/\text{GF}(2^m) \), then \( (x,y) + (x,x+y) = \infty \). The point \( (x,x+y) \), which is also a point on \( E/\text{GF}(2^m) \), is called the negative of \( P \) and is denoted by \( -P \). Also, \( -\infty = \infty \).
- Point addition and point doubling are explained in the following section.

2.8.2 Elliptic Curve Point Addition and Doubling

Consider the non-supersingular \( E/\text{GF}(2^m) \) and two points \( P = (x_1,y_1) \in E \) and \( Q = (x_2,y_2) \in E \). Let \( P \neq \pm Q \) then point addition is defined by \( P + Q = (x_3,y_3) \) and the sum \( (x_3,y_3) \in E \) is obtained by the following equations [36].

\[
\lambda = \frac{y_2 + y_1}{x_2 + x_1} \\
x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad (2.6) \\
y_3 = \lambda(x_1 + x_3) + x_3 + y_1
\]

Consider the non-supersingular \( E/\text{GF}(2^m) \) and a point \( P = (x_1,y_1) \in E \). Let \( P \neq -P \) then point doubling is defined by \( 2P = (x_3,y_3) \) and the double of \( P \), which is also a point on the curve is obtained by the following equations [36].

\[
\lambda = x_1 + \frac{y_1}{x_1} \\
x_3 = \lambda^2 + \lambda + x = x_1^2 + \frac{b}{x_1^2} \quad (2.7) \\
y_3 = x_1^2 + \lambda x_3 + x_3
\]
Different coordinate systems such as affine, standard projective, Jacobian projective, and L’opez-Dahab projective have been introduced for representing points on the non-supersingular $E/\text{GF}(2^m)$. Equations (2.6) and (2.7) present point addition and point doubling computations in affine coordinate system. Point operations presented above for the non-supersingular $E/\text{GF}(2^m)$ are performed by finite field arithmetic operations. Coordinate systems affect the complexity of point addition and point doubling operations. For instance, the complexity of point operations in affine and Jacobian projective coordinate systems in terms of the number of finite field operations is presented in Table 2.3 [35]. As presented in Table 2.3 finite field multiplication is the major finite field operation required for computation of point addition and point doubling.

Table 2.3: Complexity of point operations for the non-supersingular $E/\text{GF}(2^m)$: $y^2 + xy = x^3 + ax^2 + b$ in terms of required finite field operations (Multiplication (Mult.), Squaring (Sq.), and Inversion (Inv.) [35]

<table>
<thead>
<tr>
<th>Coordinate system</th>
<th>Point addition</th>
<th>Point doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affine</td>
<td>1 Inv.+2 Mult.+1 Sq.</td>
<td>1 Inv. + 2 Mult. + 1 Sq.</td>
</tr>
<tr>
<td>Jacobian projective</td>
<td>15 Mult. + 5 Sq.</td>
<td>5 Mult + 5 Sq.</td>
</tr>
</tbody>
</table>

2.8.3 Point Multiplication

Point multiplication, also called scalar multiplication, is a main operation in all elliptic curve security algorithms. Consider $/\text{GF}(2^m)$, an integer $k$ and two distinct elliptic curve points $P$ and $Q$. Point multiplication is defined by equation (2.8). Point multiplication can be calculated by performing point addition and point doubling repeatedly.

$$Q = kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$  \hspace{1cm} (2.8)

Knowing $E/\text{GF}(2^m)$, $P$ and $Q$, finding integer $k$ is called EC discrete logarithm problem (ECDLP). Security of ECC algorithms depend on the hardness of ECDLP. Selection of EC and its parameters is important to provide a secure ECC algorithm.

For instance, field size of $2^m$ is selected according to the security level required to be provided by the EC cryptosystem. As the field size gets larger the higher level of security can be obtained in the expense of more complex cryptosystem. In applications
with critical information larger field sizes or larger values of $m$ must be used. As shown in Table 2.4, an EC cryptosystem based on $GF(2^m)$ with $m$ greater than or equal to 233 cannot be broken till 2030. Standard sizes recommended by NIST for federal government use, are presented in Table 2.4. Recommendations of standard sizes reinforce the interoperability [37]. These sizes also achieve more efficient implementations while they provide the required security levels [37].

Table 2.4: Fields and irreducible polynomials recommended by NIST and their security lifetime

<table>
<thead>
<tr>
<th>Field</th>
<th>Irreducible Polynomial</th>
<th>Security Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GF(2^{163})$</td>
<td>$P(x) = x^{163} + x^7 + x^6 + x^3 + 1$</td>
<td>Through 2010</td>
</tr>
<tr>
<td>$GF(2^{233})$</td>
<td>$P(x) = x^{233} + x^{74} + 1$</td>
<td>Through 2030</td>
</tr>
<tr>
<td>$GF(2^{283})$</td>
<td>$P(x) = x^{283} + x^{12} + x^7 + x^5 + 1$</td>
<td>Beyond 2030</td>
</tr>
<tr>
<td>$GF(2^{409})$</td>
<td>$P(x) = x^{409} + x^{87} + 1$</td>
<td>Beyond 2030</td>
</tr>
<tr>
<td>$GF(2^{571})$</td>
<td>$P(x) = x^{571} + x^{10} + x^5 + x^2 + 1$</td>
<td>Beyond 2030</td>
</tr>
</tbody>
</table>

2.8.4 Elliptic curve based cryptographic protocols

Point multiplication is a major operation in EC based cryptographic protocols. EC Diffie–Hellman (ECDH) key exchange scheme, EC Integrated Encryption Scheme (ECIES), and EC Digital Signature Algorithm (ECDSA) are EC cryptography protocols. To present the role of point multiplication in EC cryptographic protocols, we explain EC Diffie–Hellman (ECDH) key exchange scheme as an example.

ECDH key exchange protocol is used to exchange secret key between two parties. The secret shared key can be used in symmetric key cryptography for communication. At first, domain parameters such as the type of elliptic curve, $E/GF(2^m)$, $m$, $P(x)$, the primitive point on the curve, $P$, are defined (we assume elliptic curves over binary extension fields). The domain parameters are used by two parties for communication. Figure 2.1 shows how the ECDH key exchange protocol works. Alice and Bob select their private keys $a$ and $b$, respectively (private keys are large integers). Then, Alice and Bob generate their public keys, $A$ and $B$, respectively by using their corresponding private keys and performing point multiplication operation and they exchange their public keys. The secret shared key, $T_{AB}$ is computed by the second point multiplication as
can be seen in Figure 2.1. Equation (2.9) proves that the secret key $T_{AB}$ is the same for both Alice and Bob.

$$k_{prA} = a \in \{2, 3, ..., \#E - 1\}$$
$$k_{pubA} = aP = A = (x_A, y_A)$$
$$aB = T_{AB}$$

$$k_{prB} = b \in \{2, 3, ..., \#E - 1\}$$
$$k_{pubB} = bP = B = (x_B, y_B)$$
$$bA = T_{AB}$$

Figure 2.1: ECDH key exchange protocol [1].

$$aB = a(bP) = b(aP) = bA = T_{AB} \quad (2.9)$$

From the short introduction of ECDH key exchange protocol we can see that point multiplication is the main operation in the ECDH key exchange protocol.

Figure 2.2 illustrates the hierarchical design flow of an EC cryptosystem over binary extension fields. As shown in Figure 2.2, at the top layer there exist ECC protocols such as EC Diffie–Hellman (ECDH) key exchange, EC Digital Signature Algorithm (ECDSA), and EC Integrated Encryption Scheme (ECIES). ECC protocols mainly work by utilizing point multiplication which is shown on the lower level. Point multiplication is computed by point additions and point doublings. Finally, point operations are performed using binary extension field operations. As can be seen in Figure 2.2 finite field arithmetic forms the foundation of an EC cryptosystem. Thus, efficient design of finite field operations has significant impact on the performance and efficiency of the EC cryptosystem.
2.9 Conclusion

In this chapter we have presented mathematical definitions for finite fields and a binary extension field. Moreover, binary extension field arithmetic in polynomial basis has been presented. Among finite field operations multiplication is the most important one; therefore, an example has been given to better explain this operation. To show the crucial role of finite field multiplication in an EC cryptosystem, elliptic curves over binary extension fields that are used in ECC algorithms, elliptic curve point addition, point doubling, and point multiplication have been introduced and the design flow of an EC cryptosystem has been provided. It has been shown that finite field arithmetic is the foundation of any EC cryptosystem and finite field multiplication is the essential and the most frequently used finite field operation in all ECC algorithms.
CHAPTER 3

ARCHITECTURES FOR DIGIT-Serial POLYNOMIAL BASIS MULTIPLICATION

A large number of digit-serial architectures have been proposed in the literature for efficient implementation of PB multipliers [10], [13]–[23]. In this chapter digit-serial PB multiplication scheme is described. Then, existing digit-serial PB multipliers are reviewed.

3.1 Digit-serial PB Multiplication

Consider the binary extension field $GF(2^m)$ and the irreducible polynomial $P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_1x + p_0$ with $p_i \in GF(2)$ for $0 \leq i \leq m-1$. Let $A(x)$ and $B(x)$ be two elements of $GF(2^m)$ that are represented as follows:

$$A(x) = \sum_{i=0}^{m-1} a_i x^i, \quad a_i \in \{0,1\} \text{ for } 0 \leq i \leq m-1$$

$$B(x) = \sum_{i=0}^{m-1} b_i x^i, \quad b_i \in \{0,1\} \text{ for } 0 \leq i \leq m-1$$

Then the product of $A(x)$ and $B(x)$ denoted by $C(x)$ is given by equation (3.1).

$$C = A(x)B(x) \mod P(x) \quad (3.1)$$

Operand $A(x)$ can be divided into $l$ digits of the same size $d$ starting from least significant bit. Then $A(x)$ can be represented as follows:

$$A(x) = A_{l-1}x^{(l-1)d} + \cdots + A_2x^{2d} + A_1x^d + A_0 \quad \text{where } l = [m/d] \text{ and the digit } A_j = \sum_{i=0}^{d-1} a_{jd+i}x^i \text{ for } j = 0,1,\ldots,l-1.$$ If $m$ is not a multiple of $d$ the length of the most significant digit, $A_{l-1}$, is smaller than $d$ bits; therefore, it is padded by zeros to obtain $d$-bit length digit as follows:

$$A_{l-1} = (0,\ldots,0,0,a_{m-1},\ldots,a_{(l-1)d})$$

$d$ bits
In digit-serial multiplication all bits of $B(x)$ are accessible during one multiplication; while, only one digit of $A(x)$ is accessible in each clock cycle. The product $C(x)$ is obtained as follows [14]:

$$C(x) = A(x)B(x) \mod P(x) = \left(\sum_{i=0}^{l-1} A_i x^{id}\right)B(x) \mod P(x) \quad (3.2)$$

Based on the order in which the digits are processed two different methods for a digit-serial multiplication are available, namely, most significant digit first (MSD first) method and least significant digit first (LSD first) method. The following presents the MSD first multiplication scheme [14]:

$$C = (((...(((A_{l-1}B \mod P(x)).x^d + A_{l-2}B) \mod P(x)).x^d + \cdots)x^d + A_1B) \mod P(x)).x^d + A_0B) \mod P(x) \quad (3.3)$$

The LSD first multiplication is presented by expression (3.4) [14].

$$C = (A_0B + \sum_{i=1}^{l-1} (A_{i-1}Bx^d \mod P(x) + A_iBx^d \mod P(x))) \mod p(x) \quad (3.4)$$

### 3.2 Review of Existing Digit-Serial Multipliers

A unified digit-serial inverter/multiplier in $GF(2^m)$ has been proposed in [16]. The multiplier has been designed based on a bit-serial multiplier. The data path is shared between the inverter and multiplier modules. Therefore, it has lower area complexity compared to a finite field arithmetic unit with isolated inverter and multiplier modules.

Look-up table based digit-serial multipliers in $GF(2^m)$ have been studied in [18]. The multipliers have been modeled in VHDL and synthesized by Synopsys tools in 0.35µm CMOS technology and latency, silicon area, and power consumption have been evaluated for various values of $m$ and digit sizes from 1 to 4. A new digit-serial multiplier that uses one look-up table has also been presented in [18].

A scalable digit-serial/parallel multiplier architecture for binary extension fields $GF(2^m)$ has been presented in [19]. A new technique for degree reduction that minimizes the critical path delay has been proposed. In [20] a low latency digit-serial multiplier in $GF(2^m)$ has been presented. The proposed multiplier can be pipelined to the bit-level.

In [22], Meher proposed a LSD first multiplier with less area and less area-time complexities for multiplication specifically over $GF(2^8)$ defined by an irreducible pentanomial. An efficient degree reduction has been proposed. The digit-serial multiplier
has been optimized using the degree reduction method and an efficient finite field accumulator which is proposed in [15], [21]. The idea in [22] can be extended for digit-serial multipliers in larger field sizes and for trinomials.

In [23] a parameterized MSD first multiplier has been proposed. The architecture is based on the fixed digit size of four. The field order and the irreducible polynomial are configurable. It has been shown that the multiplier has lower area complexity compared to two systolic multipliers and a sequential parallel in/parallel out multiplier for \( m=233 \).

We should take into consideration that most of the works reported in the literature worked towards high speed and/or low area architectures. Only [14] considered optimization of power consumption. In the rest of this chapter digit-serial multipliers [13]-[15], [17] that are considered in this research are explained in detail.

### 3.2.1 Song and Parhi’s Multipliers

In [14] two low-energy digit-serial PB multipliers have been proposed. Figure 3.1 and Figure 3.2 show the architectures proposed in [14] for LSD first and MSD first PB multipliers, respectively. There are two major operations in a PB multiplication, partial product generation and accumulation, and degree reduction by modulo \( P(x) \) operation. In these architectures, the first operation has been realized by using AND-gates and a binary-tree of XOR-gates. The second operation has been realized by binary-trees of XOR-gates.

A binary-tree structure has lower power consumption and delay compared to a linear-array structure because the paths are balanced in the binary tree structure which in turn reduces unwanted signal transitions (glitches) [14], [38]. Since the degree reduction is dependent on the irreducible polynomial \( P(x) \), a class of irreducible polynomials has been proposed that eliminates the linear dependency existed in the degree reduction operation which in turn enables the usage of the binary-tree structure to reduce delay and power consumption.
Energy consumption of the digit-serial multipliers was estimated using the HEAT tool [39]. This tool was proposed and developed by Satyanarayana and Parhi in 1996. The 0.5 µm technology, which was used in [14] for experiments, is very outdated; thus, the energy and power consumption evaluations are not useful anymore. In addition, experimental results were reported for multiplication over a very small field $GF(2^{16})$; hence, the issues involved in the implementation of large multipliers such as the existence of very high fan-out nets, buffer insertions, and so on that have high impacts on multipliers’ efficiencies, are not taken into account.

### 3.2.2 Tang et al. Multiplier

In [13] a low complexity MSD first multiplier has been proposed and VLSI implemented in $GF(2^{233})$ using 0.18µm technology. Figure 3.3 shows the architecture proposed in [13].
The architecture includes three modules. Module DigitMult accepts an eight-bit length digit $A_j$ and 233-bit length operand $B$ and obtains the 233-bit length output which is computed by $(A_j \times B(x)) \mod P(x)$. Module ConstMult accepts a 233-bit length input and obtains a 233-bit length output computed by the equation: $output = (input \times x^8) \mod P(x)$. Finally, module Adder adds the outputs of DigitMult and ConstMult to realize the $(A_j \times B(x)) \mod P(x) + C_{29-j} x^8 \mod P(x)$. Since this addition is a binary extension field addition it is implemented by XOR gates.

![Figure 3.3: MSD first multiplier proposed in [13] for $m = 233$ and $d = 8$.](image)

The operation of the MSD first multiplier is as follows: Choosing $m = 233$ and $d = 8$, the most significant digit of operand $A$, denoted by $A_{29}$, contains only one bit $a_{232}$ and it is padded with seven zero bits. Thus, at the beginning of multiplication instead of processing the digit $A_{29}$, one bit $a_{232}$ is checked and if $a_{232} = 1$ then the register shown in Figure 3.3 is loaded with operand $B$ otherwise it is initialized to zero. Let the register be loaded with the input operand $B$. Operand $B$ is accessible through the 233-bit input port during the multiplication. The 8-bit input port accepts a new 8-bit digit of operand $A$, denoted by $A_j$ in Figure 3.3, every clock cycle. During the first clock cycle, $A_{28}$ is read through 8-bit input port and module DigitMult generates $A_{28} B \mod P(x)$. Module ConstMult generates $B x^8 \mod P(x)$. Hence, module Adder produces $B x^8 \mod P(x) +$
\[ A_{28}B \mod P(x) \]. During the second clock cycle, module *DigitMult* generates \( A_{27}B \mod P(x) \) and module *Adder* generates \((Bx^8 \mod P(x) + A_{28}B \mod P(x))x^8 \mod P(x) + A_{27}B \mod P(x)\). During the final clock cycle module *Adder* generates the product of \( A \) and \( B \), which is denoted by \( C \), as follows: 

\[
C = AB \mod P(x) = (\cdots ((Bx^8 \mod P(x) + A_{28}B \mod P(x))x^8 \mod P(x) + A_{27}B \mod P(x))x^8 \mod P(x) + \cdots + A_1B \mod P(x))x^0 \mod P(x) + A_0B \mod P(x).
\]

### 3.2.3 Meher’s Multiplier

Typically accumulation of partial products, in a multiplier over \( GF(2^m) \), is performed by \( m \) number of 2-input XOR gates and \( m \) number of D flip-flops using feedback loops. In [15] Meher proposed a new method for the accumulation in which XOR gates and D flip-flops have been replaced by T flip-flops and feedback loops have been eliminated. Therefore, \( m \) number of XOR gates have been saved in the new accumulation unit.

However, at the beginning of a finite field multiplication, resetting T flip-flops is essential which leads to the area overhead and it can also result in timing overhead. It should be pointed out that D flip-flops have advantage over T flip-flops since there is no necessity to reset the D flip-flops.

In [15] architecture level complexity analysis has been performed to estimate the area and time complexities of the proposed multiplier [15] and similar existing multipliers by using the data from the 0.18\( \mu \)m TSMC library (no implementation has been reported). We have implemented this multiplier [15] to evaluate its power consumption and give more realistic estimation of area and time complexities.

Figure 3.4 illustrates the LSD first multiplier proposed in [15]. The multiplier architecture has two \( m \)-bit registers and a main module which is called Product Generator Cum Modular Reduction (PGCMR). One register consists of \( m \) D flip-flops and the other register which is an output register is made of \( m \) T flip-flops. At the beginning of the multiplication operation, \( m \) D flip-flops are loaded with the input operand \( B \) and the output register is initialized to zero. Module PGCMR accepts a \( d \)-bit digit of operand \( A \) and a \( m \)-bit signal \( B^{j-1} \). Signal \( B^j \) is generated by PGCMR during the clock cycle \( j \) as follows: \( B^0 = B, B^j = B^{j-1}x^d \mod P(x) \). Module PGCMR also preforms \( A_jB^j \mod P(x) \) during the clock cycle \( j \). The output register, that consists of \( m \) number
of $T$ flip-flops, preforms $C = \sum_{j=0}^{[m/d]-1} A_j B^j \mod P(x)$. The XOR operation required for this accumulation can be performed by the $T$ flip-flops because the output of a $T$ flip-flop is complemented when its input bit is set to ‘1’.

![Diagram of multiplier](image)

Figure 3.4: LSD first multiplier proposed in [15].

### 3.2.4 Kumar et al.’s Multiplier

Two different architectures referred to as Double Accumulator Multiplier (DAM) and N-Accumulator Multiplier (NAM) which are based on the LSD first multiplier in [14] have been proposed in [17]. To reduce the critical path delay of the multiplier architecture in [14], registers have been added to store the intermediate results. Compared to the LSD first multiplier in [14] they are faster. However, the area complexities and power consumption of DAM and NAM are significantly higher than the LSD first multiplier in [14].

### 3.3 Conclusion

In this dissertation a low power digit-serial PB multiplier is proposed. To present its power efficiency we should compare the proposed multiplier with the reviewed multipliers in terms of power consumption. However, some of the reviewed multipliers are with power estimation with different CMOS technologies and different design parameters and some of the reviewed works used theoretical analysis to estimate the hardware complexities and they did not report power consumption. On the other hand,
power consumption estimation at architecture level is not precise and realistic enough, because power dissipation depends on several parameters such as capacitances in the circuit and switching activities of all circuit nodes which in turn depends on input data, and this information is not available at architecture level. Therefore, to estimate power consumption of the finite field multipliers, we have implemented the reviewed multipliers [13]-[15], [17] using the same implementation configuration and evaluated their power and energy consumptions with the same power estimation methodology.
CHAPTER 4

POWER ESTIMATION IN ASIC AND LOW POWER DESIGN TECHNIQUES

To estimate the power consumption of digit-serial PB multipliers, we have implemented them using standard-cell based ASIC with static CMOS technology. Static CMOS logic style is widely used in digital system designs. For instance, Intel had a transition from domino logic to static CMOS logic to save power in Corei7 (Nehalem) series of processors [40]. In the rest of this dissertation the term "static CMOS" is simply referred to as "CMOS". In this chapter, standard-cell based ASIC design flow is introduced; sources of power dissipation in standard-cell based CMOS circuits and power calculation in standard-cell based ASIC design flow are explained. Moreover, several power reduction techniques are described.

4.1 Standard-cell based ASIC design flow

Standard-cell based ASIC design flow presents the steps that are required to convert a design specification to a chip layout. The flow requires two essential components, an EDA tool and a standard cell library which is provided by ASIC vendors. Synopsys [41] and Cadence [42] are two popular companies that provide EDA tools, and Taiwan Semiconductor Manufacturing Company (TSMC) [43] and STMicroelectronics (STM) [44] are two well-known manufacturing companies that supply standard cell libraries. The libraries contain detailed information about the characterization of the logic blocks and they also contain the wire models and they are compatible with EDA tools.

The design flow is divided into two flows called front-end and back-end. These flows will be explained in the following sections. Figure 4.1 shows the standard-cell based ASIC design flow.
4.1.1 Front-end Design Flow

As presented in Figure 4.1 front-end design flow includes logic synthesis, register transfer level (RTL) and gate level verification, and gate level optimization. Logic synthesis is the process of converting a design specification into a gate level netlist. The design is described at behavioral or RTL using a hardware description language (HDL) such as VHDL, Verilog, etc.

Logic synthesizer reads in the HDL code, the standard cell library and the design constraints at first. Then, it translates the HDL code to logic blocks that are contained in a technology independent library. Finally, the synthesizer maps the design to a particular technology library and during the mapping it optimizes the design according to the design constraints that are obtained by the designer.

Figure 4.1: Standard-cell based ASIC design flow
In this work, VHDL is used for design description. The logic synthesizer that is available through the university is provided by Synopsys and it is called Design Compiler (DC). The standard cell libraries, that are accessible through the university and they can be read in by DC and they are characterized for power consumption, are TSMC 180nm and STM 90nm and STM 65nm CMOS technology libraries.

Design verification is very important. It is performed at RTL and gate level. For verification we used NCSim from Cadance [42].

4.1.2 Back-end Design Flow
In back-end design flow the gate level netlist generated by the logic synthesizer is converted into a design layout. The cells are placed and routed by an EDA tool. Cadence Encounter can be used for back-end design flow.

4.2 Power Estimation in ASIC
In a low power design flow, power estimation is very important. Power estimation is more complicated than area or delay estimation because it depends not only on the circuit topology but also on the activity of the signals [45]. To understand how the power consumption of an ASIC is estimated, first we need to know the sources of power dissipation in the ASIC.

Since in the ASIC design flow, we have used the CMOS technology libraries, in this section first we explain the sources of power dissipation in a CMOS standard cell. Then we present how the power consumption of an ASIC, which contains CMOS standard cells, is computed. Finally, we review some existing low power design techniques.

4.2.1 Sources of power dissipation in a CMOS standard cell
For estimating and reducing power consumption in CMOS circuits, sources of power dissipation should be understood. Power dissipation in a CMOS cell is characterized by equation (4.1). The summation of the first two terms represents the dynamic power ($P_{Dyn}$). $P_{Dyn}$ is dissipated when the circuit is active and capacitances of the circuit are charged and discharged. The last term is referred to as leakage power ($P_{leak}$) and it is
dissipated even when there is no activity in the circuit. Three power consumption components \( P_{sw}, P_{int}, P_{leak} \) are explained in detail in the following sections.

\[
P = P_{sw} + P_{int} + P_{leak} \tag{4.1}
\]

4.2.1.1 Switching Power

Switching power \( P_{sw} \) is consumed due to the charging and discharging of the load capacitance of a cell. Expression (4.2) characterizes the switching power. Switching activity denoted by \( \alpha \) is the probability of the signal transition from 0 to 1 or 1 to 0 in one clock cycle. \( C_L \) represents the load capacitance at the output of the CMOS cell. \( V_{DD} \) is the supply voltage and \( f \) is the clock frequency of the circuit.

\[
P_{SW} = \frac{1}{2} \alpha C_L V_{DD}^2 f \tag{4.2}
\]

The load capacitance at the output of the CMOS cell is computed by expression (4.3). To explain how the load capacitance \( C_L \) is characterized consider the circuit illustrated in Figure 4.2. A 2-input NAND cell drives a 2-input NOR cell. The load capacitance at the output of the NAND cell is composed of the capacitance at the output node of the NAND cell \( C_{out} \), the input capacitance of the driven cell \( C_{in} \) and the total capacitance of the wire \( C_{wire} \).

\[
C_L = C_{out} + C_{in} + C_{wire} \tag{4.3}
\]

![Figure 4.2: A 2-input NAND cell drives a 2-input NOR cell](image)

4.2.1.2 Internal Power

Internal power \( P_{int} \) is dissipated due to the transitions of the internal nodes of a cell and short circuit current. During the transition of the input signals and output signal, for a short period of time both pull-up and pull-down networks conduct. Therefore, short circuit current is drawn from the power supply during the input and output transitions.

4.2.1.3 Leakage Power

Leakage power \( P_{leak} \) is expressed by (4.4). NMOS and PMOS transistors in CMOS cells are not ideal switches; therefore, when they are off still a small amount of current is
drawn from power supply to ground. This current is called leakage current and it leaks through PMOS or NMOS transistors. The amount of leakage current mainly depends on the parameters of the transistors.

\[ P_{\text{leak}} = I_{\text{leak}} V_{\text{DD}} \]  \hspace{1cm} (4.4)

### 4.2.2 Power Consumption Calculation

Synopsys Power Compiler (PC) is a power analysis and optimization tool that optimizes power consumption at gate level [46]. In the front-end design flow, we used PC for power analysis and power optimization at gate level. In this section, we explain how the power consumption of an ASIC is computed by PC. PC calculates power consumption of the design by using the formulas that are explained in this section and the information that are provided in the standard cell library.

#### 4.2.2.1 Switching Power Calculation

PC calculates switching power by equation (4.5) [46]. The output of a CMOS cell \( i \) is referred to as \( \text{net}(i) \). Since the design is made up of CMOS standard cells that create a large number of nets in the design, the total switching power is computed by summing switching power over all the nets in the design. \( C_{Li} \) represents the load capacitance at the output of a CMOS cell \( i \) (\( \text{net}(i) \)). PC uses the information from the standard cell library to determine the magnitude of \( C_{Li} \). The toggle rate, \( TR_i \), represents the number of transitions per second for \( \text{net}(i) \). Estimation of toggle rate or switching activity of a circuit is very complicated and at the same time the switching activity information has significant impact on the accuracy of the estimated power consumption. How to obtain switching activity information will be discussed later in this chapter. Supply voltage and clock frequency of the design are specified by designer based on the system application.

\[ P_{SW_{\text{total}}} = \frac{1}{2} V_{DD}^2 \sum_{\text{nets}(i)} C_{Li} TR_i \]  \hspace{1cm} (4.5)

#### 4.2.2.2 Internal Power Calculation

The internal power is calculated by using information from the standard cell library that provides internal energy for each cell in the form of lookup table and the switching activity or toggle rates of the input and output pins of the cell that are obtained by the
designer. Internal energy from each input pin of a cell is characterized in the cell library according to two situations [47]:

- When the transition of the input pin does not cause the switching on the output pin
- When the transition of the input pin causes the switching on the output pin

### 4.2.2.3 Leakage Power Calculation

To obtain the total leakage power of a design, the leakage power of all the CMOS cells existed in the design are summed up. Expression (4.6) [46] characterizes the total leakage power of a design. $P_{\text{leak}_i}$ represents the leakage power of the cell $i$. $P_{\text{leak}_i}$ is extracted from the standard cell library. $P_{\text{leak}_i}$ is state dependent which means the states of the inputs of the cell $i$ affect $P_{\text{leak}_i}$.

$$P_{\text{leak\_total}} = \sum_{\forall \text{cells}(i)} P_{\text{leak}_i} \quad (4.6)$$

### 4.2.2.4 Switching Activity Information

As mentioned earlier, switching activity information has high impact on the accuracy of power estimation. It also affects the power optimization. Switching activity of a design which is used by PC is obtained through two parameters called static probability and toggle rate [46].

Static probability and toggle rate of each net in the design are usually generated by simulation of the design [46]. Following parameters affect the static probability and the toggle rates of nets which in turn affect the result of power consumption evaluation:

- Test vectors
- Number of simulation runs
- Delay model

To provide more realistic power values, it is important to use realistic test vectors. In some applications there is a pattern for inputs. Therefore, the designer uses the pattern to generate the test vectors. In applications that there is no pattern for the inputs and the number of inputs is very large that we cannot test all the possible input transitions, random test vectors should be generated. Moreover, number of simulation runs (number of random test vectors) affect the evaluated power values.
Delay model which is used in simulation at gate level has impact on the switching activity information. Three types of simulation exist at gate level based on the delay models that are used for gates, namely zero delay, unit delay, and full-timing simulations. In zero delay simulation it is assumed that the output of each gate in the design changes instantaneously after the transition of the inputs. Thus, glitches do not occur in the zero delay simulation which results in less realistic situation for power estimation; however, zero delay simulation is faster compared to two other types of simulation [48].

In unit delay simulation it is assumed that each gate has a delay that is one unit of time. Although glitches are captured in unit delay simulation because the timing information for gates is not accurate the power consumption can be overestimated or underestimated [49]. Full-timing simulation provides the most realistic and accurate power consumption estimates. The reason is that the delay of each gate, which is characterized by the manufacturer, is captured from the standard cell library and glitches are taken into account. Delay data of the gates in the gate level netlist must be saved in a standard delay format (SDF) file [50] which is used later by simulators to perform full-timing simulation.

4.3 Low Power Design Techniques

Power consumption of a VLSI CMOS circuit can be reduced by minimizing dynamic power or leakage power. Dynamic power has a huge impact on the total power consumption. In this section, several low power techniques that mainly affect dynamic power are presented. More detailed discussion of low power design techniques can be found in [38], [51].

Dynamic power can be reduced by reducing switching power or internal power. Switching power is proportional to the square of the supply voltage, clock frequency, switching activity and node capacitances of the circuit. Internal power depends on the internal energy of each cell in the design and it is also correlated positively to the switching activity. Therefore, minimizing these parameters results in switching power minimization or internal power reduction and in turn reduces the dynamic power consumption of the CMOS circuit.
4.3.1 Voltage Scaling Techniques

Keeping all parameters constant, reducing supply voltage causes significant switching power reduction because switching power decreases quadratically with the supply voltage reduction. However, decreasing supply voltage increases the delay and degrades performance of the circuit. Several low power techniques can be applied to maintain the throughput regardless of the supply voltage reduction. These techniques are explained in the following sections.

4.3.1.1 Multi-Threshold CMOS circuit [38][51]

If threshold voltage decreases as supply voltage is reduced, the delay does not change. Therefore, we can minimize switching power by reducing supply voltage and threshold voltage accordingly without increasing the delay. Downscaling of the threshold voltage causes the problem of increase in subthreshold leakage power and decrease in noise margin. Multi-Threshold CMOS (MTCMOS) circuit can be used for reducing subthreshold leakage power.

High threshold voltage (High-Vt) and low threshold voltage (Low-Vt) transistors are used in MTCMOS circuits. High-Vt transistors consume less leakage power, but they are slower. On the other hand Low-Vt transistors consume higher leakage power; however, they are faster. Therefore, Low-Vt transistors are applied to the part of the circuit that represents the critical path and determines the circuit speed; while High-Vt transistors are used in non-critical path.

4.3.1.2 Pipelining

Pipelining is used to reduce the critical path delay. Critical path of a circuit can be divided into shorter paths by adding pipeline stages. This means that the clock frequency of the circuit can be increased. This can help to reduce the dynamic power by decreasing the supply voltage while maintaining the throughput of the circuit. However, this technique poses area and latency overheads. Therefore, it is not effective for all types of architectures and applications.
4.3.1.3 Parallelization

Consider a logic block that works with the clock frequency of $f$. This block can be replicated and more inputs can be processed and more outputs can be produced with the clock frequency of $f$. To maintain the throughput of the circuit, we can slow down each block by reducing supply voltage which causes dynamic power reduction. Similar to pipelining, this technique has area overheads.

4.3.2 Switching Minimization Techniques

Switching activity can be minimized at different design abstraction levels such as algorithm level, architecture level, gate level or circuit level. Several switching activity reduction techniques are presented in this section.

At algorithm level, data coding can achieve the lower switching activity. In applications where input data is sequential, gray coding results in less switching activity compared to binary representation [38]. Another technique for reducing switching activity is to reduce the number of operations. Usually algorithm with lower number of operations leads to an architecture with lower switching activity.

Path balancing and clock gating are two techniques at architecture level that can reduce switching activity. In logic circuits with logic depth greater than one when input signals propagate through paths with unbalanced delays, glitches happen that increase the switching activity. Therefore, balancing the delay paths eliminates glitches and reduces switching activity. Figure 4.3 shows a logic circuit composed of three XOR gates and accepts four inputs. Figure 4.3 (a) presents a linear array structure and Figure 4.3 (b) illustrates a tree structure that both realize the same function. Glitches are eliminated in the tree structure due to the equal path delays from input signals to the output as can be seen in Figure 4.3 (b). The tree structure also has smaller propagation delay compared to the linear array structure.
Figure 4.3: Realization of function f using (a) linear array structure (b) tree structure [38].

Clock gating works based on the idea that in some logic circuits for a period of time the value of the register does not change, this means that we can disable the clock of the register and reduce the switching activity which is caused by clocking the register. There is an overhead of control logic for managing the clock signal. This technique is effective for circuits with large number of registers that the values of those registers remain constant for a significant number of clock cycles.

Operand isolation is similar to clock gating technique, but it is used to disable a logic block in a system rather than disabling the register. When the output of a logic block is not used for a significant number of clock cycles, the inputs to the logic block can be kept unchanged. Thus, the logic block will be shut off for the idle clock cycles. This technique requires isolation logic and control signals for shutting off the logic block; therefore, there are timing, power and area overheads due to the control signals and isolation logic. Thus, to apply this technique careful analysis is required to guarantee that the power saving outweighs timing and area overheads.

Factoring, also called factorization, technique can be used at architecture or gate levels to reduce switching activity in the circuit. This technique is used to derive an equivalent realization of a given circuit that has lower switching activity and consumes lower switching power. It reduces switching activity of the circuit by reducing the logic depth which is connected to the nets with high switching activity [51].

Figure 4.4 shows an example of using this technique to reduce the switching activity of a circuit at gate level. The circuits shown in Figure 4.4 are two equivalent realizations of function $f = ab + cb + cd$. Assume that input ‘b’ has higher switching activity compared to the three other inputs. In Figure 4.4 (a) input ‘b’ with high switching
activity propagates through two gates at the first stage and causes higher switching activity in gates and their output nodes at the next stages. While, in the circuit illustrated in Figure 4.4 (b) input ‘b’ propagates through one gate at the second stage and it does not affect the shaded gates and their outputs at the first stage. Thus, the circuit illustrated in Figure 4.4 (b) has lower switching power and internal power compared to the circuit presented in Figure 4.4 (a).

![Diagram](image)

Figure 4.4: Factoring technique for switching activity reduction (a) implementation of the circuit (b) implementation of the circuit with reduced switching activity [51].

### 4.3.3 Switched Capacitance Minimization Technique

Gate size affects the size of a switched capacitance. Making a gate smaller reduces parasitic, internal, and load capacitances. Therefore, a smaller gate consumes lower dynamic power compared to a larger one. However, a smaller gate has lower driving strength.

Also between two gates with different functionality and identical switching activity, the gate which is smaller has smaller switched capacitances and consume lower dynamic power. For instance, a 2-input NAND gate and a 2-input AND gate have identical switching probability on their output; however, in a CMOS standard cell library a 2-input NAND gate is smaller than a 2-input AND gate and it consumes lower dynamic power. Thus, in CMOS technology replacing AND gates with NAND gates wherever it is possible leads to dynamic power reduction. We call this technique logic gate substitution. Figure 4.5 shows an example that NAND gates can be used instead of AND gates while the functionality remains unchanged. The reason is that the inputs of a 2-input XOR gate can be negated without changing the XOR output. This can be proven by expressions (4.7) and (4.8). \(f_{(a)}\) and \(f_{(b)}\) are the functions implemented by the circuits shown in Figure 4.5 (a) and (b), respectively.

\[
f_{(a)} = ab \oplus cd = ab(cd) + (ab)cd \tag{4.7}
\]
\[ f_{(b)} = \overline{ab} \oplus \overline{cd} = \overline{ab} (\overline{cd}) + (\overline{ab})cd = (\overline{ab})cd + ab(cd) \]  \hspace{1cm} (4.8)

\[ \Rightarrow f_{(a)} = f_{(b)} \]

Figure 4.5: Using logic gate substitution to reduce dynamic power (a) implementation of \( f_{(a)} \) (b) implementation of \( f_{(b)} \)

### 4.4 Conclusion

In this chapter, we have discussed the standard-cell based ASIC design flow which has been used for implementing the digit-serial PB finite field multipliers. We also have explained how the power consumption is estimated in the standard-cell based ASIC design flow. Furthermore, several low power techniques that can be used to reduce the dynamic power of digital designs have been presented.
CHAPTER 5
POWER ESTIMATION AND OPTIMIZATION FLOW

In the standard-cell based ASIC design flow that has been used as an implementation methodology, power consumption can be estimated at different levels of design abstraction. As mentioned in Chapter 4 several factors affect the accuracy of the power estimation. In this chapter, power estimation and optimization flow that is used in this research is explained.

Power analysis and optimization, especially at the gate level and lower levels of design abstraction, is tedious and complicated for most entry level designers [52]. It requires several iterations of simulation and synthesis to generate and apply switching activity and it is also difficult to setup the various EDA tools [52]. In this chapter we also discuss an automated power estimation and optimization flow that uses DC and NCSim and eliminates the laborious tasks of setting design parameters and configuring the EDA tools and thus significantly reduces the development time.

5.1 Power Optimization Flow
Besides the low power techniques that can be applied by designer, EDA tools are also used in practice for power optimization. In addition, power optimization is integrated with power estimation. The EDA tool uses power consumption as a cost function and searches for the solution that leads to the minimum power consumption. The tool requires estimating the power consumption for each solution in order to find the circuit with the lowest power consumption. As explained, accurate power estimation is required for ideal power optimization.
5.2 Power Estimation Flow

To estimate power consumption of the designs precisely, switching activity information (toggle rates and static probabilities) is obtained by full-timing simulation at gate level. In this section standard-cell based ASIC power estimation flow at gate level is explained.

At gate level, all design and implementation parameters that affect power consumption, except switching activity information, are contained in the gate level netlist and the standard-cell library [52]. Switching activity of the circuit is the function of the circuit’s input patterns and topology [53]. In order to obtain more accurate switching activity information for all the nets in the circuit, the circuit must be simulated at the gate level. Designer in turn must provide a suitable number of test inputs for the circuit based on its application and functionality in order to generate accurate and realistic switching activity information [52] as mentioned in Chapter 4. In addition, to capture glitches and obtain accurate switching activity information full-timing rather than zero delay or unit delay simulation must be performed.

In this work statistically independent random test vectors have been used because in EC cryptography no patterns exist in the input data sequences of the finite field multiplier. One thousand pairs of test vectors have been used for simulation. To perform the full-timing simulation, delay models have been extracted from the standard-cell library by DC and stored in a SDF file. Gate delays have been annotated to the gate level netlist during simulation to generate accurate switching activity information. The random test vectors and the golden outputs have been obtained by Maple\(^1\). Finite field PB multipliers have been simulated by NCSim simulator using the test vectors generated by Maple, and outputs of the multipliers have been compared with the golden outputs for verification.

To estimate power accurately, switching activity information generated by simulators must be passed to power analysis tools. Switching activity interchange format (SAIF) file is an ASCII file that contains toggle counts of each net in the design netlist, and presents for how long a signal has been at state 0 or 1 during the simulation. SAIF file is generated by some simulators and used to pass the switching activity information

\(^1\) Maple is a commercial software from Maplesoft company that analyzes, visualizes and solve mathematical problems.
to DC which performs power analysis by PC. The following symbols are used in a SAIF file to present the switching activity information for each net [51]:

- **T1**: time spent in state 1
- **T0**: time spent in state 0
- **TX**: time spent in unknown state X
- **TZ**: time spent in floating state Z
- **TB**: time spent in bus-contention state (two or more drivers driving the same object simultaneously)
- **TC**: Toggle Count
- **TG**: number of Transport Glitches. These are glitches on the output where the pulse width of the output is greater than gate delay. These consume the same power as a full transition.
- **IG**: number of Inertial Glitches. These are glitches where the pulse width of the output is smaller than gate delay. These glitches do not consume the same power as full transition, and a derating factor is used for power dissipation calculation for these glitches.

In this research SAIF file has been obtained by NCSim after 1000 simulation runs and modified so that it can be read back into DC properly. Figure 5.1 illustrates the gate level power estimation flow which is integrated into the synthesis, gate level simulation, and verification flows.

### 5.3 Automation of Power Estimation and Optimization Flow

Currently EDA tools provide power analysis and optimization tools; however, performing power analysis and optimization is a tedious and repetitive task that generally requires significant human effort in order to generate useful data. This process becomes even more troublesome when EDA tools from different vendors are used. Commands need to be executed in several iterations in order to estimate and optimize the power consumption of a single design. For comparing several designs, the tasks become even more cumbersome, which increases the probability of human error and reduces the human efficiency. [52]
5.3.1 Scripting Languages [52]
Scripting languages are high level languages that can be used to control and extend other applications. They have been used here to control two EDA tools (Synopsys DC and Cadence NCSim) which have been used in our design flow and they can improve the inter-operability between the tools from different vendors.

Most EDA tools are equipped with user interfaces that the designer uses interactively to implement the design. However, over time and development iteration, the design tasks become repetitive, tedious, and time-consuming for a designer to perform.
the tasks interactively. As such, the designer begins to make mistakes which are found much later on, or sometimes never at all. Most of the EDA tools include some level of batch scripting languages that can be used to automate these repetitive design tasks. For instance, tool command language (Tcl), which is a scripting language, can be used by DC [54], and most other EDA tools.

In the design flow that uses different EDA tools, large files or data are required to be transferred between the EDA tools. Although the file formats are standardized, they generally need to be modified with small subtle change in order to be fully compatible with the other one. Modification of large files can be easily automated using scripting languages. Moreover, EDA tools are usually executed several times in a design flow. A scripting language can be used to control the execution function of EDA tools. Shell scripting languages can be used to perform above mentioned tasks.

5.3.2 Development of Automated Power Estimation and Optimization Flow [52]

Power estimation and optimization flow has been automated by a series of programs written with scripting languages. The designs have been placed in separate folders (named by their design) which contain all necessary design files as well as specific settings files which detail such as the speed of the design, files in the design, the top level module, number of simulation cycles, golden output, etc. The scripts read these files to setup the synthesis and simulation parameters as specific information could be different between the designs under comparison.

Two main shell scripts called “run.syn” (for synthesis) and “run.sim” (for simulation) accept two arguments, the design folder (design name) and design level. When “run.syn” is called with a design level of 0, an RTL synthesis is performed using DC to generate the first gate level design using a Tcl script called “auto0.syn”. Design levels of 1 and above call “auton.syn” which reads in the switching activity from the simulation run, performs a power analysis, modifies the power requirements (reducing it by a set amount) based on the analysis results, and then performs an incremental synthesis. Both of these scripts generate and save unique reports and design files for that level of design for later viewing and analysis. When “run.sim” is called, a delay-annotated gate level simulation is performed on the indicated design level. The results are
verified with a golden solution to ensure the synthesis produced a valid gate-level netlist. Lastly the SAIF files are modified so that they can be read back into DC properly.

A designer can control DC to generate a more efficient design, by providing suitable information. DC can reduce the dynamic power consumption of a design at gate level by using low power gate level techniques if dynamic power is set as a design constraint. Expression (5.1) presents the computation of maximum dynamic power for $i$th level of synthesis. $P_{\text{maxdyn}}(i)$ represents maximum dynamic power which is used as a design constraint for level $i$. Reduction factor, $r$, in “auton.syn” accepts values from zero to less than one and it can be set by a designer. For instance, to reduce the dynamic power by 15%, reduction factor, $r$, is set to 0.85. $P_{\text{dyn}}(i - 1)$ is the estimated dynamic power for the gate level netlist which is generated by level $(i - 1)$ of synthesis.

$$P_{\text{maxdyn}}(i) = r \times P_{\text{dyn}}(i - 1) \quad (5.1)$$

Figure 5.2 shows the automated power estimation and optimization flow which performs an initial synthesis and improves the gate level design generated by initial synthesis, with two subsequent simulation and synthesis runs, considering dynamic power as a design constraint. To obtain more power efficient designs, the number of simulations and synthesis runs can be increased. More elaborate shell scripts can be created which call the necessary levels of “run.syn” and “run.sim” using various levels of parallelism which can utilize more CPU cores, or more computational workstations.

Figure 5.2: Automated power estimation and optimization flow with initial synthesis and two levels of synthesis and simulation runs [52].

5.4 Experimental results

To demonstrate the effectiveness of the proposed automated power estimation and optimization flow, four existing digit-serial PB finite field multipliers [13]-[15] have been
implemented as a case study\textsuperscript{2}. The multipliers have been implemented over $GF(2^{233})$. Trinomial $x^{233} + x^{74} + 1$, recommended by NIST, has been selected as an irreducible polynomial for generating the field. A digit size of eight has been chosen for implementation of the multipliers. The designs have been tested using 1000 pairs of input vectors (each input vector has 233 bits). The multipliers have been implemented using a low power 65nm and a general purpose 90nm standard cell libraries. [52]

In the automated design flow, the speed requirement of the system is set by a designer based on the intended application. In ECC applications, high speed finite field multipliers are usually used in EC cryptosystems that are used in internet or e-commerce servers; while, low speed finite field multipliers can be used in resource-constrained devices such as smart cards. According to our experiments, for high speed applications four finite field multiplier designs can perform at 1.25GHz clock frequency when a general purpose 90nm CMOS standard-cell library is used, and can perform at 1GHz clock frequency when a low power 65nm CMOS standard-cell library is used. An example of a low speed device is a smart card that uses the fundamental clock frequency of 13.56MHz [55]. For this case study, the finite field multipliers have been synthesized with the clock frequency of 13.56MHz. [52]

For this case study, each design had ten levels of synthesis/simulation performed. For each finite field multiplier, among the ten synthesized designs, the one that consumes the lowest amount of dynamic power has been selected as the power efficient architecture. Table 5.1 and Table 5.2 present the area, total power consumption, and energy per multiplication for the finite field multipliers in the 65nm and 90nm technologies, respectively. [52]

Table 5.1: Hardware parameters of finite field multipliers in 65nm [52]

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Area ($\mu$m$^2$)</th>
<th>Total Power (mW)</th>
<th>Energy per Mult. (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13.56 MHz</td>
<td>1 GHz</td>
<td>13.56 MHz</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>15428.9</td>
<td>16351.4</td>
<td>0.115</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>17022.2</td>
<td>18467.3</td>
<td>0.145</td>
</tr>
<tr>
<td>[14] (LSD first)</td>
<td>20251.4</td>
<td>25682.8</td>
<td>0.258</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>18450.1</td>
<td>25191.4</td>
<td>0.236</td>
</tr>
</tbody>
</table>

\textsuperscript{2} In practice the output of the multipliers are required to be saved in registers; therefore, output registers are added to the original multiplier designs in [14]. In [13], the multiplier’s output was read from a field adder module; while in this work the output is read from the register.
Table 5.2: Hardware parameters of finite field multipliers in 90nm [52]

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Area ($\mu m^2$)</th>
<th>Total Power (mW)</th>
<th>Energy per Mult. (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13.56 MHz</td>
<td>1.25 GHz</td>
<td>13.56 MHz</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>29537.5</td>
<td>29343.2</td>
<td>0.318</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>32447.3</td>
<td>32026.9</td>
<td>0.360</td>
</tr>
<tr>
<td>[14] (LSD first)</td>
<td>37346.9</td>
<td>39833</td>
<td>0.564</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>34438.3</td>
<td>37078</td>
<td>0.548</td>
</tr>
</tbody>
</table>

Using the automated design flow the results for one technology, with the four separate designs, and each design with ten levels of synthesis/simulation plus an initial synthesis (44 synthesis and 40 simulation runs) were generated in 1 hour and 9 minutes using a single core on a multi core system (3 GHz Intel Q9650 Core2 Quad). Since each synthesis/simulation task was performed sequentially, there was no idle time or interference by a designer in between the tasks. Using a non-automated flow the same amount of work in an ideal situation (assuming no human errors such as incorrect sequence of commands, or typographical errors are made) is estimated to require 8 hours and 20 minutes because of the significant human workload.

5.5 Conclusion [52]

The proposed power optimization flow results in power efficient designs in very little time. Automation of the synthesis, simulation, and power estimation and optimization flows eliminates the time-consuming and repetitive tasks that are typically performed by entry level designers. Therefore, the effects of a large number of design and implementation parameters can be examined in a shorter period of time with less difficulty and human errors. This work is intended to be helpful guide for new designers who work in the area of low power digital design and encourages them to use the scripting abilities available in the EDA tools. The proposed automated low power design flow can be extended to include the place and route stage, which may be considered as future work.
CHAPTER 6

POWER EFFICIENCY OF DIGIT-SERIAL POLYNOMIAL BASIS MULTIPLIERS IN $GF(2^{283})$

Digit-serial PB multipliers [13]-[15] that have been reviewed in Chapter 3 cannot be compared in terms of their power and energy consumption. The multipliers in [13] and [14] are with power estimation with different VLSI technologies for different field sizes which makes it difficult to compare their power efficiency. In [15] there is only a rough estimation of certain IC properties (area and speed) using data from 0.18µm TSMC library without going through actual VLSI simulation. In this work, we have performed VLSI simulation for these digit-serial PB multipliers in the same field, $GF(2^{283})$, and with the same 0.18µm VLSI technology so that an effective comparison of their power efficiency along with other IC features such as area and critical path delay can be made [56]. Recommendations of the most efficient finite field multiplier are given for the different application constraints [56]. Detailed discussion is provided for power constrained mobile and wireless applications [56]. The comparison results obtained in this research are expected to be useful for those who design and/or implement ECC for wireless and portable systems [56]. In this chapter, polynomial basis multiplication in $GF(2^{283})$ is explained. VLSI settings, comparison results and recommendations are also provided.
6.1 Polynomial Basis Multiplication in $GF(2^{283})$

In digit-serial PB multipliers [13], [15], $mod\ P(x)$ operation has been realized assuming $P(x)$ is an irreducible trinomial. As mentioned in chapter 2 for $GF(2^{283})$ no irreducible trinomial exists; therefore, irreducible pentanomial $P(x) = x^{283} + x^{12} + x^{7} + x^{5} + 1$ has been recommended by NIST for practical use.

To implement PB multipliers [13], [15], in $(2^{283})$, $mod\ P(x)$ operation has been adapted for using $P(x)$ as an above mentioned pentanomial. For instance, in [13] two modules exist that perform $mod\ P(x)$ operation. Module $DigitMult$ performs $(A_j \times B(x))\ mod\ P(x)$. $A_j$ represents a digit of size eight and it is represented by a polynomial of degree at most seven and $B(x)$ is a polynomial of degree at most 232. Module $ConstMult$ gets a 233-bit length input $C_{29-j}$ as the variable operand and $x^8$ as the constant operand and generates a 233-bit length output calculated according to the equation: $Y = (C_{29-j} \times x^8)\ mod\ P(x)$.

As an example we present how module $ConstMult$ has been adapted for using $P(x) = x^{283} + x^{12} + x^{7} + x^{5} + 1$. Let digit size is set to 8, $C_j = \sum_{i=0}^{282} c_i^{(j)} x^i$ and $C_{35-j}x^8\ mod\ P(x) = Y = \sum_{i=0}^{282} y_i x^i$. Then

$$Y = \sum_{i=0}^{282} c_i^{(35-j)} x^{i+8}\ mod\ P(x) = \sum_{i=8}^{282} c_i^{(35-j)} x^i + \sum_{i=283}^{290} c_i^{(35-j)} x^i\ mod\ P(x)$$

$$= \sum_{i=8}^{282} c_i^{(35-j)} x^i + \sum_{i=0}^{7} c_i^{(35-j)} x^i (x^{12} + x^{7} + x^{5} + 1)\ mod\ P(x)$$

$$= \sum_{i=0}^{7} c_i^{(35-j)} x^i + \sum_{i=8}^{282} c_i^{(35-j)} x^i + \sum_{i=12}^{19} c_i^{(35-j)} x^i + \sum_{i=263}^{263} c_i^{(35-j)} x^i$$

$$+ \sum_{i=7}^{14} c_i^{(35-j)} x^i + \sum_{i=5}^{12} c_i^{(35-j)} x^i$$

From (6.1) it follows that
6.2 Architecture Level Space and Time Complexities

Usually space and time complexities are used to measure the performance of hardware in finite field arithmetic. Space complexities are represented by number of XOR and AND gates, number of registers and their sizes or number of flip-flops, and MUXs. Time complexity is determined by circuit’s total gate delay. Table 6.1 and Table 6.2 show the space and time complexities for digit-serial PB multipliers [13]-[15] reviewed in chapter 3. In these tables, m denotes operand size and k is the digit size. Complexities have been estimated with the assumption of using irreducible pentanomials.

Table 6.1: Space complexities and comparisons

<table>
<thead>
<tr>
<th>Architectures</th>
<th>#FF</th>
<th>#AND</th>
<th>#XOR</th>
<th>#MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>2m + k - 1</td>
<td>km</td>
<td>km + 7k - 4</td>
<td>m</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>m + k</td>
<td>km</td>
<td>km + 7k - 4</td>
<td>0</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>m</td>
<td>km</td>
<td>km + k(6 + 3(k - 1))/2</td>
<td>0</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>2m</td>
<td>km</td>
<td>(k - 1)m + k(6 + 3(k - 1))/2</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 6.2: Time complexities and comparisons

<table>
<thead>
<tr>
<th>Architectures</th>
<th>#Clk Cycles</th>
<th>Critical Path Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>([m/k] + 1)</td>
<td>([log_2(k + 1)]T_X + T_A + T_{DFF} + T_{MUX})</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>([m/k] + 1)</td>
<td>([log_2(k + 5)]T_X + T_A + T_{DFF})</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>([m/k])</td>
<td>((\lfloor log_2k \rfloor + 3)T_X + T_A + T_{DFF})</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>([m/k])</td>
<td>((\lfloor log_2k \rfloor + 2)T_X + T_A + T_{TFF})</td>
</tr>
</tbody>
</table>

6.3 Experimental Results for Comparison

6.3.1 VLSI Experimental Settings [56]

Four existing PB multiplication architectures [13]-[15] discussed in Chapter 3, have been modeled in VHDL hardware description language at register transfer level (RTL) to realize the digit-serial multipliers in \(GF(2^{283})\). The irreducible polynomial for generating the field is \(p(x) = x^{283} + x^{12} + x^7 + x^5 + 1\). There is no trinomial for \(GF(2^{283})\). The models are synthesized by Design Compiler tool from Synopsys and the target library used is TSMC 0.18\(\mu\)m CMOS technology library. The clock frequency is set to 50 MHz for all architectures.

6.3.2 VLSI Experimental Results

Table 6.3 and Table 6.4 show power consumption, area and speed of digit-serial architectures provided by the synthesis tool. The power consumption is estimated by providing identical VLSI settings for all digit-serial architectures and using the same field size. In Table 6.3 cell internal power and net switching power represent internal power and switching power, respectively. Values in the first and the second columns are summed up to generate the dynamic power which is presented in the third column [56]. The last column shows leakage power of the design. As we expected the results also show that in 0.18\(\mu\)m process and CMOS technology, dynamic power dominates leakage power (mW vs \(\mu\)W) [56].
Table 6.3: Power consumption [56]

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Cell Internal Power (mW)</th>
<th>Net Switching Power (mW)</th>
<th>Dynamic Power (mW)</th>
<th>Cell Leakage Power (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>4.2341</td>
<td>3.5544</td>
<td>7.7885</td>
<td>6.8050</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>3.3445</td>
<td>4.4784</td>
<td>7.8229</td>
<td>5.3606</td>
</tr>
</tbody>
</table>

Table 6.4: Area, critical path delay, energy consumption and energy-delay product [56]

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Area (μm²)</th>
<th>Critical Path Delay</th>
<th>Energy for One Mult. (nJ)</th>
<th>Energy-Delay Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>195712.1</td>
<td>5.59 ns</td>
<td>5.61</td>
<td>1128.9</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>137746.2</td>
<td>5.56 ns</td>
<td>5.63</td>
<td>1126.9</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>133608.2</td>
<td>3.65 ns</td>
<td>5.8</td>
<td>762.1</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>157761.2</td>
<td>5.46 ns</td>
<td>10.36</td>
<td>2036.4</td>
</tr>
</tbody>
</table>

6.3.3 Comparison and Recommendations

As shown in Table 6.3 Song & Parhi’s LSD architecture [14] consumes the lowest dynamic power, which makes it the best option for the applications that saving power is the top concern. It can also be seen that Tang’s architecture [13] has the shortest critical path delay among all other architectures and it has also the lowest energy-delay product [56]. So it is recommended that Tang’s multiplier [13] be most suitable for the applications where both energy consumption and time delay are crucial. One clear architectural novelty for Meher’s multiplier [15] is the use of T flip-flops [56]. We note the fact that the technology which we used (0.18μm CMOS technology using TSMC library) does not have T flip-flop in the library, which could be one reason that Meher’s multiplier [15] does not show better VLSI properties [56]. On the other hand, Tang’s multiplier [13] has shown better critical path delay in the VLSI experiment than in the architectural analysis [56]. One reason could be the fact that Tang’s architecture [13] has higher regularity and less communication connections, compared to Song & Parhi’s work [14]. [56]
6.4 Conclusion [56]

Four digit-serial multiplication architectures have been simulated using 0.18μm CMOS technology in $GF(2^{283})$. The simulation results are compared in terms of power consumption, speed and area. The comparison results obtained in this work could be useful for those who design and/or implement elliptic curve cryptosystem for wireless security applications. In addition, four different architectures modeled for digit-serial multiplier implementation in $GF(2^{283})$ could be reused as soft IP cores for fast implementation of a cryptographic system.
CHAPTER 7
LOW POWER DESIGN FOR A DIGIT-Serial POLYNOMIAL BASIS FINITE FIELD MULTIPLIER IN $GF(2^m)$

This chapter presents a low power design for a digit-serial PB multiplier in $GF(2^m)$. The proposed multiplier and similar existing multipliers have been implemented on standard-cell based ASIC using CMOS 65nm and 90nm technologies. Synthesis results for area, power and energy consumptions of the proposed multiplier and similar multipliers in comparison at various digit sizes are summarized.

7.1 Proposed Low Power Design of a Digit-serial Multiplier in $GF(2^m)$
In this section we present a factoring based circuit design for a digit-serial PB multiplier in $GF(2^m)$ that reduces switching power effectively. Logic gate substitution technique is also presented that reduces internal power by using gates with lower internal energy. Gate count of the proposed digit-serial PB multiplier is also reduced.

7.1.1 Proposed Digit-serial PB Multiplier in $GF(2^m)$
An architecture diagram for the proposed digit-serial PB multiplier in $GF(2^m)$ is shown in Figure 7.1. There are three modules as shown in Figure 7.1:

- $k \times m$ multiplier takes one operand $B$ of $m$-bit and the other operand $A_j$ of $k$-bit. Note that $A_j$ changes for different clock cycles $j$. Thus, it has higher switching activity compared to operand $B$. The operation computed with $k \times m$ multiplier at clock cycle $j$ can be described as the three steps in Algorithm 1.
Algorithm 1: Operations performed by $k \times m$ multiplier

**Inputs:** \( A_j = \left( a_{k-1}^{(j)}, a_{k-2}^{(j)}, ..., a_0^{(j)} \right), B = (b_{m-1}, b_{m-2}, ..., b_1, b_0) \) and \( P(x) \).

**Output:** \( C(x) = A_j(x)B(x) \mod P(x) \).

1. Compute \( a_i^{(j)}B, i = 0,1,...,k-1; \)
2. Compute \((a_i^{(j)}B)x^i \mod p(x), i = 0,1,...,k-1; \)
3. Compute \( A_jB \mod P(x) = \sum_{i=0}^{k-1} \left( a_i^{(j)}B \right)x^i \mod p(x) \).

The above three steps are respectively realized with the circuit blocks from left to right shown in Figure 7.2(a).

- **Constant Multiplier** is a constant finite field multiplier in \( GF(2^m) \), which takes \( C_{[m/k]-j-1} \) as the variable operand and \( x^k \) as the constant operand and outputs \( C_{[m/k]-j-1} x^k \mod p(x) \).

- **Field Adder** is a parallel adder in \( GF(2^m) \) and realizes \((C_{[m/k]-j-1}x^k) \mod p(x) + A_jB \mod p(x) \). This module can be built with \( m \) two-input XOR gates formed as one layer network.

![Figure 7.1: Proposed digit-serial PB multiplier in $GF(2^m)$](image)

Note that \( k \times m \) multiplier is the most complex module among the three modules. In fact, it takes majority of system complexity in terms of gate count. By experiment we also found that its power consumption is much higher than all the other modules combined\(^3\). In the following we will propose a low power design of \( k \times m \) multiplier using factoring.

\(^3\) We assume \((m,k) = (233,8)\) for both the complexity estimation and the power consumption experiment.
and logic gate substitution methods. Complexity optimization of this module is also presented.

### 7.1.2 Low Power Design Using Factoring Technique

Consider Algorithm 2 as the modified version of the algorithm for the operation by \( k \times m \) multiplier module. While in Algorithm 1 transitions of high activity input \( A_j \) are involved in all the three steps, Step 1 in Algorithm 2 \((U_i = Bx^i \mod p(x), i = 0,1,...,k-1)\) is not affected by \( A_j \) and it also does not depend on the cycle \( j \), which means there is no input data transitions involved in this step in Algorithm 2 for all the cycles \( j = 0,1,...,[m/k]-1 \).

**Algorithm 2**: Modified algorithm for \( k \times m \) multiplier with switching minimization using factoring

**Inputs:** \( A_j = (a_{k-1}^{(j)},a_{k-2}^{(j)},...,a_{0}^{(j)}) \), \( B = (b_{m-1},b_{m-2},...,b_{1},b_{0}) \) and \( P(x) \).

**Output:** \( A_j(x)B(x) \mod P(x) \).

1. Compute \( U_i = Bx^i \mod p(x), i = 0,1,...,k-1; \)
2. Compute \( V_i^{(j)} = a_i^{(j)}U_i, i = 0,1,...,k-1; \)
3. Compute \( A_jB \mod p(x) = \sum_{i=0}^{k-1} V_i^{(j)} \).

The two designs of \( k \times m \) multiplier are shown in Figure 7.2 both of which include three sub-modules: AND network, XOR network 1 and XOR network 2. The circuit shown in Figure 7.2(a) first computes \( a_i^{(j)}B, i = 0,1,...,k-1 \) in AND network, then computes \((a_i^{(j)}B)x^i \mod p(x) = V_i^{(j)} \) at XOR network 1. In this circuit input \( A_j \) that has higher switching activity compared to input \( B \) affects all three sub-modules which results in larger number of high activity nets (outputs of the shaded modules) and thus causes higher switching activity in \( k \times m \) multiplier. Factoring can be applied to decrease the switching activity of \( k \times m \) multiplier by reducing the logic depth connected to high activity input \( A_j \).

In Figure 7.2(b), the proposed design is obtained from Figure 7.2(a) by applying factoring technique. As shown in Figure 7.2 (b), the logic depth connected to input \( A_j \) is
reduced by switching the sub-modules AND network and XOR network 1. In XOR network 1, it computes $U_i = Bx^i \mod p(x), i = 1, \ldots, k - 1$. Then at AND network, $V_i^{(j)} = a_i^{(j)} U_i, i = 0, 1, \ldots, k - 1$ are obtained. As shown in Figure 7.2(b), input $A_j$ with high switching activity does not propagate through sub-module XOR network 1. Therefore, the number of nets with high switching activity is minimized which results in lower switching activity in the proposed design.

Figure 7.2: $k \times m$ multiplier: (a) without applying factoring; (b) with applied factoring method (outputs of the shaded modules indicate nets with high switching activity).

### 7.1.3 Power consumption with and without factoring

In this section we demonstrate that how much power reduction can be achieved on $k \times m$ multiplier by applying factoring method alone. Two designs of $k \times m$ multiplier module, shown in Figure 7.2, have been implemented for $(m, k) = (233, 8)$ using CORE65LPSVT standard cell library from STMicroelectronics [57]. Switching power and internal power have been estimated by Synopsys Power Compiler tool [46] at 50 MHz clock frequency and they are given in Table 7.1. It can be seen that significant
switching power consumption can be saved for this module with the factoring method based new design. Internal power is also slightly reduced with the new method.

Table 7.1: Switching power reduction with factoring method for $k \times m$ multiplier (experiment setting: $m = 233, k = 8$).

<table>
<thead>
<tr>
<th>$k \times m$ multiplier</th>
<th>Dynamic Power</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Switching Power ($\mu W$) (Relative)</td>
<td>Internal Power ($\mu W$) (Relative)</td>
</tr>
<tr>
<td>Without factoring</td>
<td>110 (120.4%)</td>
<td>111 (101.8%)</td>
</tr>
<tr>
<td>With factoring</td>
<td>91.4 (100%)</td>
<td>109 (100%)</td>
</tr>
</tbody>
</table>

7.1.4 Gate Count Reduction

Further modification can be done to reduce the gate count. In the XOR network 1 shown in Figure 7.2(b) and Figure 7.3(a), $U_i = Bx^i \mod p(x)$, is realized with the sub-blocks $CMi, \ i = 1, 2, ..., k - 1$, where $CMi$ stands for constant multiplier with the constant input $x^i$. An observation can be made that the sub-blocks $CMi$ can be replaced with $k - 1$ number of sub-blocks $CM1$ to generate $Bx^i \mod p(x), i = 1, 2, ..., k - 1$. For this replacement Step 1 in Algorithm 2 has been changed as presented by (7.1).

$$U_0 = B; U_{i+1} = U_ix \mod p(x), i = 0, 1, ..., k - 2; \quad (7.1)$$

Figure 7.3(b) shows the proposed architecture for XOR network 1 with lower gate counts that computes the above expressions. Assuming the irreducible polynomial is $p(x) = x^m + x^n + 1, \ (0 < n < m)$, the operation performed by one $CM1$ module is given by (7.2).

$$U_{i+1} = xU_i = x \sum_{l=0}^{m-1} u_l^{(i)} x^l = u_m^{(i)} + \sum_{l=1}^{m-1} u_{l-1}^{(i)} x^l + (u_{n-1}^{(i)} + u_{m-1}^{(i)}) x^n \quad (7.2)$$

$U_i$ and $U_{i+1}$ denote the input and output to the $CM1$ constant multiplier, $i = 0, 1, ..., k - 2$, respectively. It can be seen that only one XOR gate is used for realizing $CM1$, or $(k - 1)$ XOR gates are used for the XOR network 1.

In the XOR network 1 with reduced gate count, it should be noted that the $k$ products $U_i, i = 0, 1, ..., k - 1$ of constant multiplication, have to be computed in serial fashion which, while reducing the gate count, could possibly increase the critical path delay for general irreducible polynomial $p(x)$. 

---

56
Low Power Design Using Logic Gate Substitution

As can be seen in Figure 7.2(b), proposed $k \times m$ multiplier consists of AND network following immediately by XOR network 2 that is made up of binary tree of XOR gates. Now let’s take a look at a simple example of this style logic functions that is presented by equation (7.3) where two 2-input AND gates and one XOR gate are required to implement function $y$.

$$y = (x_0 \cdot x_1) \oplus (x_2 \cdot x_3) \quad (7.3)$$

In CMOS logic, a 2-input AND gate consists of 6 transistors, while a 2-input NAND gate can be made of only 4 transistors. A 2-input NAND gate has also less number of internal nodes compared to a 2-input AND gate and as a result it consumes lower internal power compared to a 2-input AND gate.

Note that inputs of the XOR gate in equation (7.3) can be negated without changing the output (see equation (7.4)) which means that logic gate substitution can be utilized to reduce internal power by replacing AND gates with NAND gates.

$$y = (x_0 \cdot x_1) \oplus (x_2 \cdot x_3) = (x_0 \cdot x_1) \oplus (x_2 \cdot x_3) \quad (7.4)$$

It can be seen from Figure 7.4 that the modified design of function $y$ (implementation of equation (7.4)) can save 4 transistors compared to the standard method (implementation of equation (7.3)). Consequently, for the proposed multiplier with digit size $k$ there are $2km$ transistor savings.

As shown in Figure 7.4 (b) NAND gates in the modified design of function $y$ has less number of internal nodes compared to AND gates that are used in the standard design illustrated in Figure 7.4 (a). As a result, the modified design of function $y$ has

---

57
lower internal power compared to the standard design. Thus, using the modified design of function $y$ in $k \times m$ multiplier reduces internal power and leaves switching power almost unchanged which result in further reduction of dynamic power. A similar idea has been used in [22] for reducing the critical path delay and area of a digit-serial PB multiplier in $GF(2^8)$. We have not included the previous work [22] in our comparison because it did not consider the power optimization of the multiplier architecture.

The proposed low power and area efficient design of $k \times m$ multiplier with applied factoring method, logic gate substitution, and gate count reduction is shown in Figure 7.5.
Figure 7.4: (a) Standard design: implementing equation (7.3) using 24 transistors (b) Modified design: implementing equation (7.4) using 20 transistors.

Figure 7.5: $k \times m$ multiplier with applied factoring, logic gate substitution and gate count reduction.

To demonstrate that how much power reduction can be achieved on $k \times m$ multiplier by applying logic gate substitution we implemented the architecture shown in Figure 7.5 using the implementation settings mentioned in Section 7.1.3. Switching power and internal power have been estimated and they are presented in Table 7.2.
According to the experimental results gate count reduction has an insignificant impact on power consumption; therefore, power saving is effectively caused by logic gate substitution. As presented in Table 7.2 internal power consumption of \( k \times m \) multiplier has been reduced significantly by applying logic gate substitution.

Table 7.2: Internal power reduction with logic gate substitution for \( k \times m \) multiplier (experiment setting: \( m = 233, \ k = 8 \)).

<table>
<thead>
<tr>
<th>( k \times m ) multiplier with factoring</th>
<th>Dynamic Power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Switching Power (( \mu W )) (Relative)</td>
</tr>
<tr>
<td>Without logic gate substitution</td>
<td>91.4 (99%)</td>
</tr>
<tr>
<td>With logic gate substitution</td>
<td>92.3 (100%)</td>
</tr>
</tbody>
</table>

### 7.1.6 Architecture Level Complexities

To evaluate the complexities of finite field arithmetic hardware, circuit complexity is usually presented by the number of flip-flops (FF), 2-input AND/NAND gates, 2-input XOR gates and MUXs. In estimation of critical path delay, \( T_A, T_{NA}, T_X, T_M, T_D, \) and \( T_T \) are used to refer to as the delay caused by a 2-input AND gate, a 2-input NAND gate, a 2-input XOR gate, a 2-to-1 multiplexer, a D flip-flop, and a T flip-flop, respectively.

As shown in Figure 7.5, the complexity of \( k \times m \) multiplier module can be estimated as follows. XOR network 1 with reduced gate counts (Figure 7.3(b)) contains \((k - 1)\) number of \( CM1 \) modules. The number of NAND gates in the NAND network are equal to the number of AND gates in the AND network. The operations involved in the AND network are shown in Step 2 in Algorithm 2, \( V_t^{(j)} = a_t^{(j)} U_t, i = 0,1, \ldots, k - 1 \), which use \( km \) AND gates. Therefore, the NAND network also contains \( km \) NAND gates. The XOR network 2 contains \( m \) binary trees each with \( k \) inputs, which requires \((k - 1) m \) XOR gates. Besides \( k \times m \) multiplier, Constant Multiplier, Field Adder and the register (Figure 7.1) require \( k \) number of XOR gates, \( m \) number of XOR gates, and \( m \) D flip-flops, respectively. Table 7.3 and Table 7.4 show the complexities for our proposed architecture and some similar digit-serial PB multipliers in the literature, where we take
$p(x)$ a trinomial. Note that in Table 7.3 output registers are added to some existing works to support a stable output\(^4\).

Table 7.3: Architecture level area complexities and comparisons

<table>
<thead>
<tr>
<th>Architectures</th>
<th>#FF</th>
<th>#AND</th>
<th>#NAND</th>
<th>#XOR</th>
<th>#MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>$3m + k - 1$</td>
<td>$km$</td>
<td>0</td>
<td>$km + 3k - 2$</td>
<td>$m$</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>$2m + k$</td>
<td>$km$</td>
<td>0</td>
<td>$km + 3k - 2$</td>
<td>0</td>
</tr>
<tr>
<td>[17] (LSD first)</td>
<td>$4m + k - 2$</td>
<td>$km$</td>
<td>0</td>
<td>$(k + 1)m + 3k - 3$</td>
<td>$m$</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>$2m$</td>
<td>$km$</td>
<td>0</td>
<td>$km + k(k + 1)/2$</td>
<td>0</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>$2m$</td>
<td>$km$</td>
<td>0</td>
<td>$(k - 1)m + k(k + 1)/2$</td>
<td>0</td>
</tr>
<tr>
<td>Proposed</td>
<td>$m$</td>
<td>0</td>
<td>$km$</td>
<td>$km + (2k - 1)$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.4: Time complexities and comparisons

<table>
<thead>
<tr>
<th>Architectures</th>
<th>#Clk cycles</th>
<th>Critical Path Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>$[m/k] + 2$</td>
<td>$[\log_2(k + 1)]T_X + T_A + T_D + T_M$</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>$[m/k] + 2$</td>
<td>$[\log_2(k + 3)]T_X + T_A + T_D$</td>
</tr>
<tr>
<td>[17] (LSD first)</td>
<td>$[m/k] + 2$</td>
<td>$[\log_2([k/2] + 1)]T_X + T_A + T_D + T_M$</td>
</tr>
<tr>
<td>[13] (MSD first)</td>
<td>$[m/k] + 1$</td>
<td>$([\log_2 k] + 2)T_X + T_A + T_D$</td>
</tr>
<tr>
<td>[15] (LSD first)</td>
<td>$[m/k] + 1$</td>
<td>$([\log_2 k] + 1)T_X + T_A + T_T$</td>
</tr>
<tr>
<td>Proposed</td>
<td>$[m/k] + 1$</td>
<td>$([\log_2 k] + 2)T_X + T_{NA} + T_D$</td>
</tr>
</tbody>
</table>

7.2 VLSI Experimental Results

7.2.1 Experimental Settings

Our proposed design and the similar existing architectures have been modeled in VHDL hardware description language at register transfer level (RTL) to realize a digit-serial PB multiplier in $GF(2^{233})$ with digit size of eight. The irreducible trinomial $p(x) = x^{233} + x^{74} + 1$ has been used for generating the field.

The designs have been synthesized by Design Compiler (DC) tool from Synopsys and STMicroelectronics CORE65LPSVT standard cell library at supply voltage of 1 V, 25°C temperature and at typical process corner has been used for synthesis. CORE65LPSVT is a library for 65nm low power CMOS digital design [57]. All designs

\(^4\) Subsequently, in Table 7.4 their number of clock cycles is also adjusted accordingly. Note that the number of clock cycles should be $[m/k]$ for $(m, k) = (233,8)$ in both the work in [13] and our proposed work.
have been synthesized with 50 MHz clock frequency. This frequency is suitable for low power devices such as smart cards.

To estimate the power consumption of the proposed digit-serial PB multiplier and similar multipliers in comparison we used the power estimation flow which has been explained in Chapter 5 and 1000 random vectors for each input A and B have been used for obtaining switching activity information.

7.2.2 Synthesis Results

Synthesis results are presented in Tables 7.5 and 7.6. As presented in Table 7.5 our proposed multiplier has the lowest switching power and internal power and thus it consumes the lowest amount of dynamic power. Compared to our proposed multiplier the best previous work [14] consumes about 38.4% higher dynamic power.

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Switching Power (µW)</th>
<th>Internal Power (µW)</th>
<th>Dynamic Power (µW)</th>
<th>Leakage Power (nW)</th>
<th>Total Power (µW) (Relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>319.97</td>
<td>626.66</td>
<td>946.63</td>
<td>949.22</td>
<td>947.58 (246.5%)</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>138.40</td>
<td>393.03</td>
<td>531.43</td>
<td>650.74</td>
<td>532.08 (138.4%)</td>
</tr>
<tr>
<td>[17]</td>
<td>350.95</td>
<td>807.16</td>
<td>1158.11</td>
<td>950.42</td>
<td>1159.06 (301.5%)</td>
</tr>
<tr>
<td>[13]</td>
<td>154.65</td>
<td>406.38</td>
<td>561.03</td>
<td>654.38</td>
<td>561.68 (146.1%)</td>
</tr>
<tr>
<td>[15]</td>
<td>355.25</td>
<td>512.49</td>
<td>867.74</td>
<td>865.42</td>
<td>868.61 (226%)</td>
</tr>
<tr>
<td>Proposed</td>
<td>127.81</td>
<td>256.16</td>
<td>383.97</td>
<td>450.10</td>
<td>384.42 (100%)</td>
</tr>
</tbody>
</table>

Table 7.6: Area, energy consumption, and energy-area product

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Area (µm²) (Relative)</th>
<th>Energy per Mult. (pJ) (Relative)</th>
<th>Energy-Area Product (J.m². 10⁻¹⁹) (Relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>20251.4 (141.4%)</td>
<td>587.52 (263.5%)</td>
<td>118.98 (372.5%)</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>17022.2 (118.8%)</td>
<td>329.9 (148%)</td>
<td>56.16 (175.8%)</td>
</tr>
<tr>
<td>[17]</td>
<td>23762.44 (165.9%)</td>
<td>718.64 (322.3%)</td>
<td>170.77 (534.7%)</td>
</tr>
<tr>
<td>[13]</td>
<td>17202.12 (120.1%)</td>
<td>325.79 (146.1%)</td>
<td>56.04 (175.5%)</td>
</tr>
<tr>
<td>[15]</td>
<td>18450.12 (128.8%)</td>
<td>538.56 (241.5%)</td>
<td>99.36 (311.1%)</td>
</tr>
<tr>
<td>Proposed</td>
<td>14323.92 (100%)</td>
<td>222.97 (100%)</td>
<td>31.94 (100%)</td>
</tr>
</tbody>
</table>

In Table 7.5 last column presents the total power consumption of the proposed digit-serial PB multiplier and the similar existing multipliers. The total power
consumption has been obtained by adding dynamic power and leakage power. However, it can be seen that leakage power is significantly smaller than dynamic power (nW vs. µW). This is the reason that we have optimized dynamic power rather than leakage power. As shown in Table 7.5 our proposed multiplier consumes the least amount of total power among all other multipliers.

Our proposed multiplier and the existing multipliers in comparison complete one field multiplication in different number of clock cycles. Therefore, it is beneficial to use energy per one multiplication as an efficiency measure for comparison because it gives equal weight to both power and computational delay. The energy for one multiplication operation for the new architecture is about 31.6% lower than the best existing result given in Table 7.6.

Table 7.6 shows a comparison of Energy-Area (EA) product between our proposed field multiplier and the similar existing multipliers. We introduce EA product metric as a comprehensive metric that takes into account the criteria such as area complexity, and energy consumption. As shown in Table 7.6, the proposed multiplier has the lowest EA product among all other multipliers. Compared to the proposed architecture the best previous work is still about 75.5% higher in the EA product. Figure 7.6 shows the energy consumed by one multiplication and the area complexity for the proposed multiplier and the existing multipliers of similar architectural style.

![Figure 7.6: Energy per multiplication vs. area](image-url)
7.3 The effect of digit size on the hardware parameters of the digit-serial PB multipliers

In this section the effect of digit size on the area complexity, power and energy consumptions of the digit-serial PB multipliers have been investigated. The proposed and similar existing digit-serial PB multipliers have been realized using different digit sizes, 8, 16, 32 and 64. The reason for choosing these digit sizes is that they present standard bus width in embedded systems. Clock frequency of 50 MHz has been selected for synthesis and power estimation. This operating frequency is suitable for smart card applications. Finite field multipliers have been synthesized for minimum dynamic power consumption. CORE65LPSVT and CORE90GPSVT standard cell libraries from STMicroelectronics at typical process corner, at 1 V supply voltage and 25°C temperature have been used for implementation.

7.3.1 The effect of digit size on area complexity

Table 7.7 and Table 7.8 show the synthesis results for the area complexity of the multipliers in 65nm and 90nm technologies, respectively. Figure 7.7 illustrates the results graphically. As digit size increases from 8 to 64 the area complexity of the proposed multiplier increases more gradually compared to the area complexity of the other multipliers in comparison. In addition, compared to the similar existing multipliers the proposed multiplier has the lowest area complexity for all digit sizes.

Table 7.7 : Area complexity of multipliers in (µm²) for various digit sizes in 65nm technology

<table>
<thead>
<tr>
<th>Architectures</th>
<th>k = 8</th>
<th>k = 16</th>
<th>k = 32</th>
<th>k = 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>20251.4</td>
<td>33268.04</td>
<td>60804.64</td>
<td>114756.72</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>17022.2</td>
<td>29446.56</td>
<td>54458.56</td>
<td>104123.24</td>
</tr>
<tr>
<td>[17]</td>
<td>23762.44</td>
<td>36805.6</td>
<td>64312.04</td>
<td>113779.12</td>
</tr>
<tr>
<td>[13]</td>
<td>17202.12</td>
<td>29665.48</td>
<td>56140.24</td>
<td>112279.96</td>
</tr>
<tr>
<td>[15]</td>
<td>18450.12</td>
<td>31781.36</td>
<td>58377.28</td>
<td>121414.8</td>
</tr>
<tr>
<td>Proposed</td>
<td>14323.92</td>
<td>25454.52</td>
<td>48427.08</td>
<td>94562</td>
</tr>
</tbody>
</table>

Table 7.8: Area complexity of multipliers in (µm²) for various digit sizes in 90nm technology

<table>
<thead>
<tr>
<th>Architectures</th>
<th>k = 8</th>
<th>k = 16</th>
<th>k = 32</th>
<th>k = 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14] (LSD first)</td>
<td>37352.43</td>
<td>61375.6</td>
<td>111394.32</td>
<td>204587.15</td>
</tr>
<tr>
<td>[14] (MSD first)</td>
<td>32447.25</td>
<td>55877.72</td>
<td>103438.92</td>
<td>203485.16</td>
</tr>
<tr>
<td>[17]</td>
<td>43631.79</td>
<td>68357.43</td>
<td>117754.92</td>
<td>214988.01</td>
</tr>
<tr>
<td>[13]</td>
<td>32862.14</td>
<td>56010.53</td>
<td>109578.89</td>
<td>219367.43</td>
</tr>
<tr>
<td>[15]</td>
<td>34443.79</td>
<td>58631.6</td>
<td>111661.04</td>
<td>218750.58</td>
</tr>
<tr>
<td>Proposed</td>
<td>28389.42</td>
<td>50419.35</td>
<td>95789.75</td>
<td>189343.68</td>
</tr>
</tbody>
</table>
7.3.2 The effect of digit size on power consumption

The power consumptions of the multipliers for different digit sizes are illustrated in Figure 7.8. For each multiplier, when the digit size increases, the number of bits that are processed in each clock cycle also increases. Therefore, the amount of work that has been done per unit of time increases which in turn results in higher power consumption as can
be seen in Figure 7.8. The proposed multiplier consumes the lowest amount of power compared to the other multipliers at all digit sizes.

An interesting point that can be derived from the results, shown in Figure 7.8, is that the power consumption difference between the multipliers is more significant for larger digit sizes. The result analyzes mentioned above, are valid for both 65nm and 90nm technologies.

![Figure 7.8: Total power consumption of the multipliers in (a) 65nm technology (b) 90nm technology](image)
7.3.3 The effect of digit size on energy consumption

Figure 7.9 (a) and (b) present energy per multiplication for the digit-serial PB multipliers in 65nm and 90nm technologies, respectively. As shown in Figure 7.9, our proposed multiplier has the lowest energy per multiplication operation for all different digit sizes in both 65nm and 90nm technologies.

Figure 7.9: Energy per multiplication for the multipliers in (a) 65nm technology (b) 90nm technology
Energy trend is different for different multipliers. First we analyze the results for 65nm technology. Our proposed multiplier consumes less energy by increasing digit size from 8 to 64. We can also, conclude that digit size of 32 is the optimal digit size for the LSD first multiplier proposed in [14], since for this digit size the multiplier consumes the lowest amount of energy. The multiplier proposed in [15], consumes the least amount of energy for digit size 16. Since energy consumption affects the battery life, by choosing appropriate digit size for each multiplier, the battery life of the system can be extended. In 90nm technology, the LSD first multiplier proposed in [14] consumes less energy by increasing digit size from 8 to 32 and its energy consumption remains almost unchanged for digit size of 64. It should be mentioned that the rest of the analyses of the results that mentioned above, are also valid for 90nm technology used for implementation as can be seen in Figure 7.9 (b).

7.4 Conclusion
Factoring technique has been adopted for a new architecture level design that minimizes the switching activities and consequently reduces the power consumption of a digit-serial PB multiplier in $GF(2^m)$. Logic gate substitution technique has also been utilized to further reduce the power consumption of the digit-serial PB multiplier. Moreover, the area complexity of the finite field multiplier has been reduced. To evaluate our proposed multiplier we have implemented it along with other similar multipliers with digit size 8 and clock frequency of 50MHz using 65nm technology. The VLSI experimental results show that our proposed architecture consumes about 27.8% less power and 31.6% less energy than the best previous work in comparison. The proposed multiplier also achieves 43% lower EA product compared to the best previous work. The proposed low power digit-serial PB multiplier is suitable for implementing low power EC cryptosystems in embedded systems with limited power resources. The proposed digit-serial PB multiplier can also be used as an IP core for fast implementation of EC cryptosystems.

The proposed multiplier along with other similar multipliers has also been implemented with digit sizes 8, 16, 32 and 64 using 65nm and 90nm technologies. Then the effects of various digit sizes on the hardware parameters such as area complexity, and
power and energy consumptions have been investigated. The proposed multiplier is the most power and energy efficient multiplier at all the digit sizes.
CHAPTER 8

CONCLUSION

8.1 Summary and Conclusion

EC cryptosystems are computationally intensive and consume a large amount of power. High power consumption reduces the reliability of the system, increases the cooling and packaging cost, and reduces battery life of the wireless battery-powered devices. Thus, power consumption reduction of an EC cryptosystem is very important.

Power consumption of an EC cryptosystem can be reduced at different levels of its design flow. At the very top level there are EC cryptography protocols such as EC Diffie–Hellman (ECDH) key agreement scheme, EC Integrated Encryption Scheme (ECIES), and EC Digital Signature Algorithm (ECDSA). Major operation in these protocols is called point (scalar) multiplication. Point multiplication is realized by point addition and point doubling on the elliptic curve points. Since an elliptic curve is defined over a finite field, point addition and point doubling are performed using finite field arithmetic. Finite field arithmetic is the foundation of the EC cryptosystem design flow. This dissertation has presented power reduction and power analysis of finite field multiplication operation that is the most important operation among finite field arithmetic operations.

Two types of finite fields that are usually used in EC cryptosystems are prime fields, $GF(p)$ and binary extension fields, $GF(2^m)$. Binary extension fields are more attractive for hardware implementation since they offer carry-free arithmetic. Therefore, they are suitable for low power wireless cryptography applications. In $GF(2^m)$,
multiplication is the most important operation because field addition is simply a bitwise XOR operation between two field elements and field inversion can be realized by consecutive finite field multiplications.

In the first chapter the scope of our research has been introduced briefly. Different methods such as PB, normal basis, and dual basis that are usually used to represent field elements in $GF(2^m)$ have been introduced. Three architecture styles that are commonly used for hardware implementation of finite field PB multipliers have also been presented. Finally, power optimization and power analysis of digit-serial PB finite field multipliers have been introduced as the scope of our research. In Chapter 2, binary extension field $GF(2^m)$, polynomial basis and binary extension field arithmetic have been explained and mathematical background of elliptic curve cryptosystems over $GF(2^m)$ is also discussed. Digit-serial finite field PB multipliers are reviewed in Chapter 3.

Power analysis of the proposed low power digit-serial finite field PB multiplier and similar multipliers in comparison cannot be performed as easy and accurate as their area and time complexity analysis at architecture level using only theoretical analysis. The multipliers have to be implemented using a hardware platform. Then, their power consumption can be estimated using EDA tools. In Chapter 4, standard-cell based ASIC design flow that is used to implement the multipliers has been introduced and power calculation by PC in this design flow has been explained.

Different parameters can affect the accuracy of the estimated power. In Chapter 5 we have explained our power estimation method and its parameters. EDA tools can be used interactively to estimate and optimize power consumption of the designs; however, power analysis especially at the gate level and lower levels of design abstraction is complicated for most entry level designers. Several iterations of synthesis and simulation runs are usually required for the synthesizer to optimize the power consumption of the design. An automated power estimation and optimization flow has been discussed. The proposed automated flow helps digital designers to estimate and optimize the power consumption of designs accurately and effectively at gate level with much less effort.

It is difficult to compare power efficiency of the digit-serial PB multipliers that have been reviewed in Chapter 3 because some of them are without power estimation and those with power estimation have been implemented using different VLSI technologies.
and different field sizes. In Chapter 6, we compared power consumption of some of the reviewed multipliers. To make an effective comparison of their power efficiency, the multipliers have been implemented using the same field $GF(2^{283})$ and VLSI technology. The comparison results could be helpful for EC cryptosystem designers to choose the most power efficient multiplier.

In Chapter 7, a new low power digit-serial PB multiplier over $GF(2^m)$ has been presented. In digit-serial PB multipliers one input operand is fixed and does not change during one multiplication while the other input may change. This means that one input has higher switching activity compared to the other input. Considering this property of the input operands of the digit-serial PB multipliers, factoring technique has been adopted to reduce the switching activity of the multiplier architecture that consequently reduces the dynamic power consumption of the multiplier. Logic gate substitution has also been utilized to further reduce the dynamic power. The automated power estimation and optimization flow has been used for synthesizing the proposed multiplier and similar digit-serial PB multipliers. It has been shown that the proposed digit-serial PB multiplier consumes significantly less power and less energy compared to the similar PB multipliers. We also implemented the proposed and similar existing finite field multipliers at digit sizes of 8, 16, 32, and 64 to find out how much is the effect of digit size on their area complexity, and power and energy consumption. The synthesis results show that the proposed multiplier has the lowest area complexity, the lowest power dissipation, and the lowest energy consumption at all the digit sizes. The proposed and the similar existing multipliers that have been implemented in this work can be reused as IP cores for fast implementation of EC cryptosystems.

8.2 Future Works
Power estimation is more complex at backend design stage. Power estimation flow for backend design stage can be automated.

Low power design of digit-serial finite field multipliers in other bases can be investigated.
Since the power consumption of the EC cryptosystem can be reduced at different levels of its design flow, low power design of ECC algorithms or point operations worth to be explored.

Further research can be conducted in designing finite field multipliers using nanoelectromechanical (NEM) relays as energy efficient devices. In addition, designing energy efficient finite field multipliers based on adiabatic logic circuits can be studied.
BIBLIOGRAPHY


APPENDIX A
COPYRIGHT PERMISSION

| Title: | An automated power estimation and optimization methodology |
| Conference: | Signals, Circuits and Systems |
| Proceedings: | (ISSCS), 2015 International Symposium on |
| Author: | Namin, S.H.; Muscato, R.; Huapeng Wu; Ahmadi, M. |
| Publisher: | IEEE |
| Date: | 9-10 July 2015 |
| Copyright © 2015, IEEE |

**Thesis / Dissertation Reuse**

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

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

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © [2011] IEEE.

2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] appear prominently with each reprinted figure and/or table.

3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

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

1) The following IEEE copyright credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

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

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

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

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

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

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

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.
Shoaleh Hashemi Namin was born in Tehran, Iran, in 1981. She received her BSc degree in Computer Engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2004, and the MSc degree in Computer Engineering from Sharif University of Technology, Tehran, Iran in 2006. Since January 2010, she has been pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, On, Canada. Her research interests include digital integrated circuits, hardware implementation of finite field multipliers and cryptography algorithms, and low power VLSI design.