NOVEL SINGLE AND HYBRID FINITE FIELD MULTIPLIERS OVER $GF(2^M)$ FOR EMERGING CRYPTOGRAPHIC SYSTEMS

by

Jiafeng Xie

M.E., Central South University, China, July 2010

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

PhD

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Jiafeng Xie

It was defended on

August 4, 2014

and approved by

Zhi-Hong Mao, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

Luis F. Chaparro, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

Thomas McDermott, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Gregory Reed, Ph.D., Professor, Department of Electrical and Computer Engineering

Ervin Sejdić, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Mingui Sun, Ph.D., Professor, Department of Neurological Surgery

Dissertation Directors: Zhi-Hong Mao, Ph.D., Associate Professor, Department of Electrical and Computer Engineering,

Pramod Kumar Meher, Ph.D., Senior Research Scientist, School of Computing Engineering, Nanyang Technological University

NOVEL SINGLE AND HYBRID
FINITE FIELD MULTIPLIERS OVER $GF(2^M)$
FOR EMERGING CRYPTOGRAPHIC SYSTEMS

Jiafeng Xie, PhD

University of Pittsburgh, 2014

With the rapid development of economic and technical progress, designers and users of various kinds of ICs and emerging embedded systems like body-embedded chips and wearable devices are increasingly facing security issues. All of these demands from customers push the cryptographic systems to be faster, more efficient, more reliable and safer. On the other hand, multiplier over $GF(2^m)$ as the most important part of these emerging cryptographic systems, is expected to be high-throughput, low-complexity, and low-latency. Fortunately, very large scale integration (VLSI) digital signal processing techniques offer great facilities to design efficient multipliers over $GF(2^m)$.

This dissertation focuses on designing novel VLSI implementation of high-throughput low-latency and low-complexity single and hybrid finite field multipliers over $GF(2^m)$ for emerging cryptographic systems. Low-latency (latency can be chosen without any restriction) high-speed pentanomial basis multipliers are presented. For the first time, the dissertation also develops three high-throughput digit-serial multipliers based on pentanomials. Then a novel realization of digit-level implementation of multipliers based on redundant basis is introduced. Finally, single and hybrid reordered normal basis bit-level and digit-level high-throughput multipliers are presented. To the author′s knowledge, this is the first time ever reported on multipliers with multiple throughput rate choices. All the proposed designs are simple and modular, therefore suitable for VLSI implementation for various emerging cryptographic systems.

TABLE OF CONTENTS

PREFACE xvi

1.0 INTRODUCTION 1

1.1 PRELIMINARY: PROBLEM STATEMENT AND MOTIVATION 2

1.2 OBJECTIVES OF THE DISSERTATION 4

1.2.1 Low latency systolic montgomery multiplier for finite field $GF(2^m)$ based on pentanomials 5

1.2.2 Low-latency high-throughput systolic multipliers over $GF(2^m)$ for NIST pentanomials 5

1.2.3 High-throughput finite field multipliers using redundant basis for FPGA and ASIC implementations 6

1.2.4 Single and hybrid architectures for multiplication over finite field $GF(2^m)$ based on reordered normal basis 7

1.3 OUTLINE OF THE DISSERTATION 7

2.0 FINITE FIELD MULTIPLICATIONS OVER $GF(2^M)$ 9

2.1 INTRODUCTION TO FINITE FIELD 9

2.1.1 Basic concepts about finite field 9

2.1.2 Field operations 10

2.1.3 Field order 10

2.1.4 Binary fields 10

2.2 POLYNOMIAL BASIS MULTIPLICATION OVER $GF(2^M)$ 11

2.2.1 Important irreducible polynomials 13

2.2.2 AOPs 13

2.2.3 Trinomials . . . 14
2.2.4 Pentanomials . . . 14
2.2.5 Existing works about polynomial basis multiplication over $GF(2^m)$ . . . 15
2.2.6 Existing works on AOPs . . . 15
2.2.7 Existing works on trinomials . . . 16
2.2.8 Existing works on pentanomials . . . 16
2.2.9 Research direction . . . 16
2.3 NORMAL BASIS MULTIPLICATION OVER $GF(2^M)$ . . . 17
2.3.1 Some properties . . . 18
2.3.2 Squaring . . . 18
2.3.3 Normal basis multiplication and its example . . . 19
2.3.4 Existing works on RB (Type-I ONB) . . . 20
2.3.5 Existing works on RNB (Type-II ONB) . . . 20
2.3.6 Research direction . . . 21
3.0 LOW LATENCY SYSTOLIC MONTGOMERY MULTIPLIER FOR FINITE FIELD $GF(2^M)$ BASED ON PENTANOMIALS . . . 22
3.1 INTRODUCTION . . . 22
3.2 ALGORITHM . . . 23
3.2.1 Montgomery multiplication . . . 24
3.2.2 PCA technique . . . 27
3.3 PROPOSED STRUCTURES . . . 31
3.4 HARDWARE AND TIME COMPLEXITY . . . 35
3.5 CONCLUSTION . . . 36
4.0 LOW-LATENCY HIGH-THROUGHPUT SYSTOLIC MULTIPLIERS OVER $GF(2^M)$ FOR NIST PENTANOMIALS . . . 38
4.1 INTRODUCTION . . . 38
4.2 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL MULTIPLIERS-I . . . 40
4.2.1 Proposed algorithm . . . 40
4.2.2 Proposed bit-parallel systolic multiplier-I (BP-I) . . . 45
4.2.3 Proposed modified BP-I (MBP-I) . . . 46

4.2.4 Proposed digit-serial systolic multiplier-I (DS-I) 49
4.3 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL SYSTOLIC MULTIPLIERS-II 50
4.3.1 Proposed algorithm 50
4.3.2 Proposed bit-parallel systolic multiplier-II (BP-II) 55
4.3.3 Proposed modified BP-II (MBP-II) 55
4.3.4 Proposed digit-serial systolic multiplier-II (DS-II) 57
4.4 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL SYSTOLIC MULTIPLIERS-III 59
4.4.1 Proposed algorithm 59
4.4.2 Proposed bit-parallel systolic multiplier-III (BP-III) 63
4.4.3 Proposed digit-serial systolic multiplier-III (DS-III) 63
4.5 AREA AND TIME COMPLEXITIES 65
4.5.1 Complexities of BP-I and DS-I 65
4.5.2 Complexities of BP-II and DS-II 66
4.5.3 Complexities of BP-III and DS-III 69
4.5.4 Comparison of area and time complexities 69
4.5.5 ASIC implementation 71
4.6 CONCLUSION 73
5.0 HIGH-THROUGHPUT FINITE FIELD MULTIPLIERS USING REDUNDANT BASIS FOR FPGA AND ASIC IMPLEMENTATIONS . 74
5.1 INTRODUCTION 74
5.2 MATHEMATICAL FORMULATION 76
5.2.1 Brief review of existing digit-serial RB multiplier 76
5.2.2 Proposed digit-serial RB multiplication algorithm 78
5.3 DERIVATION OF PROPOSED HIGH-THROUGHPUT STRUCTURES FOR RB MULTIPLIERS 81
5.3.1 Proposed structure-I 81
5.3.2 Modification of proposed structure-I 86
5.3.3 Proposed structure-II 87

5.3.4 Proposed structure-III . . . 89
5.4 AREA-TIME-POWER COMPLEXITIES . . . 90
5.4.1 Complexities of PS-I, PS-II and PS-III . . . 90
5.4.2 Comparison with existing digit-serial RB multipliers . . . 93
5.4.3 Comparison with existing digit-serial multipliers having a type I ONB . . . 93
5.4.4 Comparison of synthesis results for FPGA implementation . . . 93
5.4.5 Comparison of synthesis results for ASIC implementation . . . 96
5.4.6 Design selection . . . 100
5.5 CONCLUSION . . . 100
6.0 SINGLE AND HYBRID ARCHITECTURES FOR MULTIPLICATION OVER FINITE FIELD $GF(2^M)$ BASED ON REORDERED NORMAL BASIS . . . 103
6.1 INTRODUCTION . . . 103
6.2 PRELIMINARIES . . . 105
6.3 PROPOSED BIT- AND DIGIT-PARALLEL RNB MULTIPLIERS . . . 106
6.3.1 Proposed bit-parallel (BP) RNB multiplication algorithm . . . 106
6.3.2 Proposed BP architecture for RNB multiplier . . . 112
6.3.3 Proposed digit-parallel (DP) architecture for RNB multiplier . . . 115
6.3.4 Proposed low-latency bit- and digit-parallel (LBP) (LDP) architectures for RNB multiplier . . . 117
6.4 PROPOSED DIGIT-SERIAL (DS) RNB MULTIPLIER . . . 121
6.4.1 Brief review of existing DS RNB multiplier . . . 121
6.4.2 Proposed DS RNB multiplier . . . 122
6.5 PROPOSED HYBRID RNB MULTIPLIER . . . 127
6.5.1 Algorithm . . . 128
6.5.2 Architecture . . . 129
6.5.2.1 Architecture with two throughput choices . . . 129
6.5.2.2 Architecture with three throughput choices . . . 131
6.5.2.3 Architecture with four throughput choices . . . 132
6.6 COMPLEXITY COMPARISON . . . 134

6.6.1 Complexities of proposed MBP and DP architectures 134
6.6.2 Complexities of proposed DS architectures 135
6.6.3 Complexity of proposed hybrid architecture 137
6.6.4 Comparison of area and time complexities between proposed architectures 137
6.6.5 Comparison with existing digit-serial multipliers 137
6.6.6 FPGA implementation 140
6.6.6.1 Optimal implementation 140
6.6.6.2 Comparison of FPGA implementation 140
6.6.7 ASIC implementation 142
6.6.7.1 Optimal implementation 142
6.6.7.2 Comparison of ASIC implementation 142
6.6.8 FPGA implementation of hybrid architecture 144
6.7 CONCLUSION 144
7.0 SUMMARY AND FUTURE WORK 146
7.1 DISSERTATION SUMMARY 146
7.2 FUTURE WORK 148
BIBLIOGRAPHY 149

LIST OF TABLES

1 Hardware- and time-complexities of proposed and the existing structures for multiplication over $GF(2^m)$ based on pentanomials 34

2 Comparison of area and time complexities 36

3 Comparison of area-time complexity of bit-parallel systolic multipliers 67

4 Comparison of area-time complexity of digit-serial systolic multipliers 68

5 Comparison of area-time complexities of digit-serial RB multipliers 91

6 Comparison of area-time complexities of different multipliers where there exist a type I ONB ($n = m + 1$) 92

7 Area-time-power complexities comparison of various multipliers based on FPGA implementation 97

8 Area-time-power complexities comparison of various multipliers based on ASIC implementation 101

9 Comparison of area-time complexities of proposed RNB multipliers 136

10 Comparison of area- and time-complexities of different digit-serial multipliers where there exist a type II ONB 138

11 Comparison of area- and time-complexities of different digit-serial multipliers where there exist a type II ONB (continual) 139

LIST OF FIGURES

1 Detailed expression of M_1 and M_2. (a) The detail of M_1. (b) The detail of M_2. 28

2 An example of bit-addition involved in M_2. 28

3 Process of derivation of $A'^{(9)}$ directly from $A'^{(7)}$. 29

4 Process of PCA technique. (a) Process of derivation. (b) Process of derivation. 30

5 SFG for forward block of (3.21) for m=13. (a) SFG and cut-set retiming. (b) Functional description of node R. (c) Functional description of the ith bit-multiplication node M(i). (d) Functional description of the bit-addition node AD. 32

6 Proposed systolic-like structure for m=13. (a) Proposed structure. (b) Design of the PE-1. (c) Design of a regular PE. (d) Design of the second PE in the array (from right). (e) Structure of NMRC in the PE-1 of the 2nd array. (f) An example of structure of the AND cell. 33

7 Proposed bit-parallel systolic multiplier-I (BP-I) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of PE-1. (d) Internal structure of regular PE. (e) Internal structure of PE-d. 45

8 Proposed modified bit-parallel systolic multiplier-I (MBP-I) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of regular PE of Array-1. (d) Internal structure of regular PE of Array-2 to Array-$w-1$, where $2 \leq i \leq w-1$. (e) Internal structure of regular PE of Array-w. 48

9 Proposed digit-serial systolic multiplier-I (DS-I), where Δ denotes unit delay and black block denotes register cell. (a) Proposed structure. (b) Internal structure of PE[0], where $0 \leq u \leq w-1$. (c) Internal structure of a regular PE. (e) Internal structure of AC cell. 49

10 Proposed bit-parallel systolic multiplier-II (BP-II), where Δ denotes unit delay and black block denotes register cell $g = (2w - k1 + k3)$. (a) Proposed structure. (b) Internal structure of PRC, where M-I cell is designed into two-stage pipeline to reduce the critical-path to T_X. (c) Internal structure of a regular PE. (d) Detailed structure of M-V cell in PRC (derives $B^0_{\{P\}}$ from B^0). 54

11 Proposed modified bit-parallel systolic multiplier-II (MBP-II) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell $g = (2w - k1 + k3)$. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of regular PE of Array-1. (d) Internal structure of regular PE of Array-2 to Array-$w-1$, where $2 \leq i \leq w-1$. 56

12 Proposed digit-serial systolic multiplier-II (DS-II), where Δ denotes unit delay and black block denotes register cell $g = (2w - k1 + k3)$. (a) Proposed structure. (b) Internal structure of PE[0], where $0 \leq u \leq w-1$. (c) Internal structure of a regular PE. (d) Internal structure of AC cell. 58

13 Proposed bit-parallel systolic multiplier-III (BP-III), where Δ denotes unit delay and black block denotes register cell. (a) Proposed structure, where each gray box represents a systolic-array module. (b) Internal structure of a regular PE, where $1 \leq v \leq d-1$ (there is no XOR cell in PE-1). (c) Internal structure of SPT. (d) Internal structure of PCA.(e) Internal structure of FMA. 64

14 Proposed digit-serial systolic multiplier-III (DS-III), where Δ denotes unit delay and black block denotes register cell. 65

15 Comparison of latency of proposed and existing bit-parallel systolic multipliers. 70

16 Comparison of area-delay products (ADP)s of proposed and existing bit-parallel systolic multipliers. 70

17 Comparison of area-delay products (ADP)s of proposed and existing digit-serial multipliers. (a) Comparison of ADPs of proposed and existing structures. (b) Detailed comparison. 71

18 Comparison of average computation time (ACT) of proposed and existing digit-serial multipliers. 72

19 Signal-flow graph (SFG) for parallel realization of RB multiplication over $GF(2^m)$ based on (12) and (13). (a) The proposed SFG. (b) Functional description of S node, where S-I node performs circular bit-shifting of one position and S-II node performs circular bit-shifting by Q positions. (c) Functional description of M node. (d) Functional description of A node. 82

20 Processor-space flow graph (PSFG) of digit-serial realization of finite field RB multiplication over $GF(2^m)$. (a) The proposed PSFG. (b) Functional description of add-accumulation (AA) node. 83

21 Cut-set retiming of PSFG of finite field RB multiplication over $GF(2^m)$, where "D" denotes delay. 84

22 Proposed structure-I (PS-I) for RB multiplier, where "R" denotes a register cell. (a) Detailed structure of the RB multiplier. (b) Structure of the bit-permutation module (BPM). (c) Structure of the AND cell in the partial product generation module (PPGM). (d) Structure of the XOR cell in the PPGM. (e) Structure of the register cell in the PPGM. (f) Structure of the finite field accumulator. 85

23 PS-I for RB multiplier when $d = 2$. (a) Proposed cut-set retiming of PSFG when $d = 2$. (b) Detailed internal structure of merged regular PPGU. (c) Corresponding PS-I for the case $d = 2$. 86

24 Proposed structure-II (PS-II) for RB multiplier, where "R" denotes a register cell. (a) Modified PSFG. (b) Structure of RB multiplier. 88

25 Novel cut-set retiming of PSFG and its corresponding structure: PS-III. (a) Cut-set retiming. (b) BPM and PPGM of PS-III. 89

26 Comparisons of key metrics of various structures for $n = 268$. (a) Comparisons of area-complexity (number of ALUT). (b) Comparisons of maximum frequency (MHz). (c) Comparisons of power consumption (mW). 94

27 Comparisons of area-delay-power complexities of various structures for $n = 268$ (delay refers to the ACT of a structure. Area, delay and power are measured in number of ALUT, 10^{-4}s and mW, respectively). (a) Comparisons of area-delay product (ADP). (b) Comparisons of power-delay product (PDP). (c) Comparisons of area-delay-power product (ADPP). 95

28 Comparisons of key metrics of various structures for $n = 268$. (a) Comparisons of area-complexity (μm^2). (b) Comparisons of delay (ns). (c) Comparisons of power consumption (mW). 98

29 Comparisons of area-delay-power complexities of various structures for $n = 268$ (delay refers to the ACT of a structure. Area, delay and power are measured in μm^2, ns and mW/GHz, respectively). (a) Comparisons of area-delay product (ADP). (b) Comparisons of power-delay product (PDP). (c) Comparisons of area-delay-power product (ADPP). 99

30 Extended expression of equation (6.6), where the bits in the dashed box are extra added bits. 108

31 Proposed bit-parallel (BP) RNB multiplication architecture, where $\triangle$ stands for unit delay. (a) Proposed architecture. (b) Function of the first PE. (c) Function of the regular PE (PE[2] to PE[$2m$]). 111

32 Proposed modified bit-parallel-I (MBP-I) RNB multiplication architecture, where $\triangle$ stands for unit delay. (a) Modified architecture. (b) Internal structure of parallel adder arrays. 113

33 Detailed design of proposed MBP-II multiplier, where "D" denotes a register cell and $\triangle$ stands for unit delay. (a) Design of proposed architecture. (b) Design of BPM. (c) Structure of the AND cell. (d) Structure of the XOR cell. (e) Structure of the register cell. 114

34 Proposed digit-parallel (DP) architecture for $d = 2$. (a) Two regular PEs of Fig. 31. (b) Merged PE. (c) Internal structure of two regular PEs of Fig. 31. (d) Internal structure of merged PE. (e) Design detail of proposed RNB multiplier for $d = 2$. . . 116
35 Proposed low-latency bit-parallel (LBP) architecture for RNB multiplier. 119
36 Proposed LBP architecture for $d = 2$. 120
37 Proposed digit-serial (DS) architecture for RNB multiplier. (a) Proposed DS architecture. (b) Function of AA cell. 124
38 Proposed DS architecture for RNB multiplier for $T = 2/Q$. 126
39 Proposed hybrid architecture for RNB multiplier with two throughput choices. (a) Proposed hybrid architecture. (b) Functional description of demultiplexer (DM) cell. 130
40 Proposed hybrid architecture for RNB multiplier with three throughput choices. . . 131
41 Proposed hybrid architecture for RNB multiplier with four throughput choices. . . 132
42 Design diagram for proposed hybrid RNB multiplier. (a) Proposed diagram. (b) Functional description of selector signals. 133
43 Area-delay product (ADP) trend along with d for proposed MBP-I. Unit: number of ALUT$\cdot 10^{-6}$. 141
44 Area-delay product (ADP) comparison of proposed and existing structures. Unit: number of ALUT$\cdot 10^{-6}s$. 141
45 Power-delay product (PDP) comparison of proposed and existing structures. Unit: mW$\cdot 10^{-6}s$. 142
46 Area-delay product (ADP) trend along with d for proposed MBP-I. Unit: um$^2\cdot$ns. . 143
47 Area-delay product (ADP) comparison of proposed and existing structures. Unit: um$^2\cdot$ns. 143
48 Chip-view of proposed hybrid architecture in FPGA device. 144

PREFACE

I would like to give my highest gratitude to my advisor, Professor Zhi-Hong Mao, for his support, encouragement, and guidance that I can go through my Ph.D. study at the University of Pittsburgh peacefully and successfully. Many thanks also to my co-advisor, Dr. Pramod Kumar Meher, for his diligent help and warm care.

I would like to thank Professor Luis F. Chaparro, Professor Thomas McDermott, Professor Gregory Reed, Professor Ervin Sejdić and Professor Mingui Sun for their service as my Ph.D. committee members.

Many thanks to all the members of my research group, especially Shimeng Huang and Qinhao Zhang, for their kind help in my Ph.D. study.

I also would like to thank all my friends here in Pittsburgh, especially Andy Wang and his family, Tom Piccone and his family, Williams, Yohan, Silver, Shasha, Jerry, Timothy Feng and his family, Po-yu and his family...all the members in Church in Pittsburgh. Many thanks to Prof. Jung Han, Department of Electrical Engineering, Yale University.

Endless thanks to my parents, my parents in law, my lovely wife, Jane Ni, and my younger brother and sister, for their great love, constant support, and warm encouragement.

1.0 INTRODUCTION

Cryptographic engineering can be traced back about 2000 years ago, when two parties are required to share a symmetric key for encryption and decryption for communications. As technology advances, the ancient cryptographic technologies are no longer applicable, e.g., for the approached suggested 2000 years ago, two parities must met each other and agree on the symmetric key. Modern cryptographic techniques have been developed a lot since 1976, when two scientists Die and Hellman invented an algorithm which leads to today's public key cryptography systems [1-5]. Recently, technologies like wearable and portable devices, deeply embedded systems, wireless sensor nodes, RFID tags develops in a significant speed and greatly changes people's daily life. Meanwhile, security issues arise with all these devices, i.e., how to maintain the data processed in a correct form, how to protect the data from stolen and so on. All these challenges require advanced or emerging cryptographic systems to meet the critical/specific requirements. It is worth mentioning that all these cryptographic systems need efficient realization of finite field multiplication over $GF(2^m)$. The multiplication over $GF(2^m)$ is much more complex than the addition operation in finite fields. Besides, the multiplication operation can be extended further to perform division, exponentiation, and inversion [6]-[8]. Generally, the multiplication over $GF(2^m)$ can be easily implemented on a general purpose machine. But the software implementations usually do not meet the , speed, area, cost constrains and some environmental requirements for specific embedded systems. Most of the real-time embedded systems, therefore, need hardware implementation of finite field arithmetic operations. Luckily, modern very large scale integration (VLSI) provides us opportunities to deal with these issues. All these topics can be viewed as an applied science in the overlap between mathematics, computer engineering and electrical engineering.

1.1 PRELIMINARY: PROBLEM STATEMENT AND MOTIVATION

Security issues arise significantly in various resource-constrained environments (such as deeply embedded systems, smart cards, portable devices, and wearable devices) and high-performance web server (such as online banking systems) [1]-[5]. Undoubtedly, all these applications require highly efficient cryptographic computing systems, such as elliptic curve cryptography (ECC). On one hand, the resource-constrained applications suffer from availability of resources like chip area. On the other hand, these systems also suffer from low throughput ability of the current cryptographic systems. Moreover, due to increasing number of small and connected devices to the internet servers, efficient computation of cryptographic systems are crucial for every devices.

There are three major features determine the performance of a certain cryptographic system: speed, area occupation and safety. Meanwhile, as more and more technologies exposed to the public, it becomes very challenging for modern cryptographic systems to handle those increasing heavy tasks. Therefore, researchers are still working toward the safer and faster cryptographic systems.

Finite field multiplication over $GF(2^m)$ is widely used in emerging cryptographic systems as a basic component, e.g., ECC and error control coding systems [1]-[5]. The multiplication over $GF(2^m)$ is much more complex than the addition operation in finite fields. Besides, the multiplication operation can be extended further to perform division, exponentiation, and inversion [6]-[8]. In general, there are three major basis to perform the multiplication over $GF(2^m)$, i.e., polynomial basis, normal basis and redundant basis. All the three basis can be used in various application environments due to their different characteristics and usually multiplication over certain basis will mostly be used because of its simplicity and regularity of the structure.

The choice of basis to represent field elements can determine the performance of the field arithmetic [9]-[11]. Modern cryptographic systems usually involve a number of arithmetic operations such as the multiplication, squaring and so on. It is worthy mentioning that normal basis multipliers have no hardware cost in squaring operations (only involve bits-shifting), more and more cryptographic circuits designs prefer to use the normal basis

multipliers compared with the multipliers based on the other two bases. There are two special classes of normal basis for which the complexity of multiplication can be minimized, namely the optimal normal basis (ONB) type I and II. ONB Type I and II has been widely used for various cryptographic system designs [12]-[19].

On the other side, however, polynomial basis multipliers have advantages over normal basis multipliers such as modularity and regularity of the multiplier, easy to be realized in high-throughput systems. The National Institute of Standards and Technology (NIST) has also recommended five binary finite fields for ECC implementation [8], and two of those are generated by the trinomials, $f(x) = x^{233} + x^{73} + 1$ and $f(x) = x^{409} + x^{87} + 1$ [8], while the other three are pentanomials. Therefore, a number of works have been done on efficient realization of multiplication over $GF(2^m)$ based on various basis [13]-[16].

These works can be classified into two types, in terms of the design style, the systolic designs and the non-systolic designs. The non-systolic designs may have low-latency, but yield low throughput, while the systolic designs feature modularity, regularity and local interconnections, which are important properties for VLSI design [20]-[25], and all the processing elements (PEs) in a systolic array are fully pipelined to produce very high throughput rate [26]-[34]. But, on the other hand, systolic structures usually suffer from large number of registers in structure. In terms of the input/output style, the existing works can be classified as bit-serial, digit-serial, bit-parallel, digit-parallel and serial/parallel hybrid structures [35]-[41]. In real application environment, however, considering the tradeoff between the area and delay complexity of a design, most researches are focused on the low-speed implementations [42]-[54]. Therefore, there is great potential on high-throughput, low-latency and low-complexity realization of finite field multiplication over $GF(2^m)$ [55]-[67].

The systolic implementation of finite field multiplication over $GF(2^m)$ usually has benefits like high-throughput, low critical-path and stable output. But on the other side, it suffers long latency and large number of registers used in the structure for data transferring. Systolic structure usually needs a large number of registers in the PEs, which greatly increases the overall area of the systolic structure. Therefore, we should find an efficient way to reduce the register count. Moreover, since the latency of systolic structure is long, efforts should be made to reduce the latency of the whole structure. Furthermore, great efforts

should be made on the efficient digit-serial structure which not only has high-throughput ability but also bring benefits to the area reduction.

VLSI digital signal processing (DSP) design techniques are widely used in various application systems [20]. It is believed that with proper VLSI DSP design techniques, practical high-throughput low-complexity digit-serial implementation of finite field multiplication over $GF(2^m)$ can be obtained. This thesis is devoted to develop a bottom-up approach for feasible low-complexity high-throughput and low-latency multipliers over $GF(2^m)$ for emerging cryptographic systems.

1.2 OBJECTIVES OF THE DISSERTATION

In this dissertation, novel VLSI implementation of high-throughput, low-latency and low-complexity finite field multipliers over $GF(2^m)$ for emerging cryptographic systems are presented. A bottom-up approach is proposed in designing these efficient multipliers. Trinomial based multiplier has been thoroughly studies in past years because of its characteristics like simplicity and regularity of the multiplication process. In this thesis, we thus only focus on normal basis and pentanomial based multipliers. Normal basis multipliers usually are complex and once they are not preferred for cryptographic systems because of their complicate multiplication process. But through years' derivation, some reordered versions, like redundant basis (RB) and reordered normal basis (RNB) [68], merged that the normal basis multipliers have great potential in cryptographic systems especially because of its free squaring operations. On the other hand, low-latency high-throughput bit-level and digit-level multipliers based on pentanomials are rarely reported due to their complicated multiplication, though three of them are recommended by NIST for real applications.

Thus, in this dissertation, we firstly propose novel bit-level and digit-level implementation of multipliers based on RB (type I ONB). Novel single and hybrid architectures for RNB (type II ONB) multipliers are also presented. High-throughput Low-latency bit- and digit-level pentanomial basis multipliers are discussed and presented. The objectives of this report are to design high performance and fast finite field multipliers for emerging/advanced

cryptographic systems for small and wearable devices based on different security level and key size.

The major contributions of this dissertation are presented in the following:

1.2.1 Low latency systolic montgomery multiplier for finite field $GF(2^m)$ based on pentanomials

In this chapter, a low latency bit-parallel systolic Montgomery multiplier over $GF(2^m)$ based on irreducible pentanomials. We have presented an efficient algorithm to decompose the multiplication into a number of independent units for parallel processing. Besides, we have introduced a novel so-called pre-computed addition (PCA) technique to further reduce the latency. We have also proposed a novel modular reduction (NMR) operation based on the PCA technique, which is more suitable for deriving low latency multiplier compared to modular reduction schemes in existing multipliers. Moreover, by suitable cut-set retiming, we have derived here a low latency bit-level-pipelined systolic design for field multiplication based on our proposed algorithm. The proposed design has the same critical-path as the corresponding existing design, but offers at least one-fourth of the latency of the other. For the pentanomial suggested by NIST, the proposed design offers 74% reduction of area-delay product over the recently reported design.

1.2.2 Low-latency high-throughput systolic multipliers over $\boldsymbol{GF(2^m)}$ for NIST pentanomials

Recently, finite field multipliers with capabilities like low-latency and high-throughput have gained great attention in emerging cryptographic systems. Systolic realization of low-latency and high-throughput multipliers over $GF(2^m)$ for NIST pentanomials, however, are not so abundant. In this chapter, we present three pairs of low-latency and high-throughput bit-parallel and digit-serial systolic multipliers specifically based on NIST pentanomials. Novel decomposition-technique has been proposed that the multiplier is decomposed into several parallel processing arrays to obtain a bit-parallel systolic structure (BP-I) with a critical-path of $2T_X$ (T_X is the propagation delay of an XOR gate). These parallel arrays are

then projected along vertical direction to obtain a digit-serial structure (DS-I) with the same critical-path. To increase the throughput rate, another pair of bit-parallel (BP-II) and digit-serial (DS-II) structures are then presented based on a novel modular reduction operation, where the critical-paths are reduced to $T_A + T_X$ (T_A is the propagation delay of an AND gate). Identical data sharing between processing elements (PEs) has been proposed to reduce area-complexity of BP-I and BP-II further. Finally, we have proposed Karatsuba Algorithm (KA)-based bit-parallel (BP-III) and digit-serial (DS-III) multipliers to enhance the throughput rate further. From synthesis results, it is shown that the proposed multipliers have significantly lower latency and higher throughput than the existing designs. To the authors' knowledge, this is the first report on low-latency systolic multipliers based on NIST pentanomials without any restriction on latency choice.

1.2.3 High-throughput finite field multipliers using redundant basis for FPGA and ASIC implementations

RB multipliers over $GF(2^m)$ have gained huge popularity in ECC mainly because of their negligible hardware cost for squaring and modular reduction. In this chapter, we have proposed a novel recursive decomposition algorithm for RB multiplication to obtain high-throughput digit-serial implementation. Through efficient projection of signal-flow graph (SFG) of the proposed algorithm, a highly regular processor-space flow-graph (PSFG) is derived. By identifying suitable cut-sets, we have modified the PSFG suitably and performed efficient feed-forward cut-set retiming to derive three novel multipliers which not only involve significantly less time-complexity than the existing ones but also require less area and less power consumption compared with the others. Both theoretical analysis and synthesis results confirm the efficiency of proposed multipliers over the existing ones. The synthesis results for field programmable gate array (FPGA) and application specific integrated circuit (ASIC) realization of the proposed designs and competing existing designs are compared. It is shown that the proposed high-throughput structures are the best among the corresponding designs, for FPGA and ASIC implementation. It is shown that the proposed designs can achieve up to 94% and 60% savings of area-delay-power product (ADPP) on FPGA and ASIC

implementation over the best of the existing designs, respectively. The proposed designs, therefore, can be used in various application environments.

1.2.4 Single and hybrid architectures for multiplication over finite field $GF(2^m)$ based on reordered normal basis

RNB, a reordered version of an ONB II, has great potential to be used in modern/emerging cryptographic systems because of its efficient realization in multiplication and squaring operations over $GF(2^m)$. In this chapter, efficient bit- and digit-level algorithms for computing multiplication over $GF(2^m)$ based on RNB are presented. Novel high-throughput low-complexity architectures are presented based on these proposed algorithms. First of all, high-throughput bit- and digit-parallel multipliers are presented. To have an optimal balanced trade-off between area and time complexities, novel digit-serial architectures for RNB multiplication is proposed then. Finally, for the first time, a novel hybrid architecture for parallel/serial realization of finite field multiplication based on RNB is introduced. The main advantage of the novel hybrid architecture is that it offers flexible choices of throughput of parallel/serial realization of RNB multiplication while meantime it involves little hardware overhead. This feature would be a major advantage for implementing multiplication in modern/emerging reconfigurable cryptographic systems. Both theoretical comparison and practical simulation results from FPGA and ASIC realization are presented. It is shown that the proposed multipliers have significantly lower area-time-power complexity than the corresponding existing designs. Specifically, FPGA realization of the novel hybrid multiplier is detailed presented to confirm its efficiency in FPGA based reconfigurable cryptographic platforms.

1.3 OUTLINE OF THE DISSERTATION

The rest of this dissertation is organized as follows. In total, we have four major chapters covering the technical problems.

Chapter 2 gives a brief review of mathematic formulation of polynomial basis and normal basis multiplication over $GF(2^m)$. Several classes of the most frequently used basis multipliers are also introduced. Moreover, the previous reported design styles of finite field multipliers over $GF(2^m)$ are also presented.

Chapter 3 introduces a low latency bit-parallel systolic Montgomery multiplier over $GF(2^m)$ based on irreducible pentanomials. This is the first report specifically focus on pentanomial basis multipliers.

Chapter 4 proposes three pairs of low-latency and high-throughput bit-parallel and digit-serial systolic multipliers specifically based on NIST pentanomials. This is the first report on low-latency systolic multipliers based on NIST pentanomials without any restriction on latency choice.

Chapter 5 introduces the design of serveral high-throughput multipliers over $GF(2^m)$ based on RB and their implementations in both FPGA and ASIC platforms. The design procedure of the multiplications are proposed as well as the novel architectures.

Chapter 6 presents several novel high-throughput low-complexity architectures based on RNB. Bit- and digit-parallel, digit-serial and hybrid architectures are presented, respectively, as well as their performance in both FPGA and ASIC platforms.

Finally, Chapter 7 gives a summary of the contributions of the entire dissertation and provides future research directions.

2.0 FINITE FIELD MULTIPLICATIONS OVER $GF(2^M)$

In this chapter, we present a brief review of mathematic background of finite field multiplication over $GF(2^m)$. Some basic concepts of the finite field operations are introduced, including addition and multiplication. Several classes of the most frequently used irreducible polynomials are also described. Moreover, the previous reported designs of polynomial basis multiplication over $GF(2^m)$ are given.

2.1 INTRODUCTION TO FINITE FIELD

The efficient implementation of finite field arithmetic is critical to modern cryptographic systems like the elliptic curve systems because the operations involved in these systems are performed using arithmetic operations in theses fields.

2.1.1 Basic concepts about finite field

Fields, expressed as F, are generally defined as abstractions of familiar number systems (such as the real numbers R) as well as their essential properties. Usually, two operations, namely addition (denoted by $+$) and multiplication (denoted by $\cdot$) are involved in the field, which satisfy the usual arithmetic properties, such as the distributive law: $(a+b)\cdot c = a\cdot c + b\cdot c$ for all $a,\ b,\ c \in F$ [1]-[5]. If the field contains finite elements, then the field is called to be finite.

2.1.2 Field operations

Two basic operations are involved in the field arithmetic operations, *i.e.*, addition and multiplication. Subtraction can be expressed in terms of addition: for $a,\ b \in field\ F,\ a - b = a + (-b)$, where $-b$ is the unique element in the field that $b + (-b) = 0$. Similarly, division of field elements can be expressed by multiplication: for $a,\ b \in field\ F$ with $b \neq 0$, we have $a/b = a \cdot b^{-1}$ [1]-[5].

2.1.3 Field order

The order of a certain finite field is determined by the number of elements in that field. For a finite field F, if the order of this field is q and q is a prime power, i.e., there exists $q = pm$ where p is a prime number and m is a positive integer. If $m = 1$, then this field is defined as a prime field. If $m \geq 2$, then this field is defined as an extension field [1]-[5].

2.1.4 Binary fields

If the order of a finite field F is 2^m, this field is called binary field or characteristic-two field, expressed as $GF(2^m)$. Polynomial basis representation can be used to construct $GF(2^m)$, where the elements of $GF(2^m)$ are binary polynomials (coefficients $\in \{0,\ 1\}$, and degree $\leq m - 1$). The elements in the finite field $GF(2^m)$ are the polynomials as $\{0,\ 1,\ \alpha,\ \alpha + 1,\ \alpha^2,\ \alpha^2 + 1, \ldots,\ \alpha^{m-1} + \alpha^{m-2} + \ldots + \alpha + 1\}$, where α is the root of irreducible polynomial $f(x)$ over $GF(2)(GF(2)=\{0,\ 1\})$, i.e., $f(\alpha) = 0$ [1]-[5].

The irreducible binary polynomial is defined as follows: For a polynomial of degree m, this polynomial cannot be factored as a product of binary polynomials with degree less than m, this polynomial is called the irreducible polynomial.

The major arithmetic operations in the finite field $GF(2^m)$ are the addition and multiplication. Addition in field $GF(2^m)$ is simple and easy, namely the addition of polynomials, with coefficient arithmetic performed modulo 2, which can be implemented by XOR gates in hardware design. On the other side, the multiplication is much more complex, and it is performed by field elements modulo the reduction polynomial. For example, for any binary

polynomial $a(x)$, $a(x)\text{mod}f(x)$ denotes the reminder polynomial of degree less than m. And this operation is called reduction modulo $f(x)$ [1]-[5].

2.2 POLYNOMIAL BASIS MULTIPLICATION OVER $GF(2^M)$

Recently, the binary field $GF(2^m)$ has gained substantial interests due to its widely usage in various applications, such as algebraic codes, cryptographic systems, random number generators, VLSI DSP systems and VLSI testing platforms. Among all these applications, the multiplication over $GF(2^m)$ is served as a common computing core and can be extended further to obtain the operations of division, exponentiation, and inversion. Thus, in this section, we present the basic steps of the polynomial basis multiplication over $GF(2^m)$ as follows:

Let $f(x)$ be a degree m irreducible polynomial over $GF(2)$ in the form [1]-[5], [36]

$$f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + f_0 \tag{2.1}$$

where $f_i \in GF(2) = \{0,\ 1\}$. Then the set $\{1,\ x, \ldots, x^{m-1}\}$ is the polynomial basis in $GF(2^m)$ defined by $f(x)$ as

$$a(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 \tag{2.2}$$

where $a_i \in GF(2)$.

Let $a(x)$ and $b(x)$ be two field elements and $c(x)$ be the product, then

$$c(x) = a(x)b(x)\text{mod } f(x) \tag{2.3}$$

where

$$b(x) = b_{m-1}x^{m-1} + \cdots + b_1 x + b_0 \tag{2.4}$$

$$c(x) = c_{m-1}x^{m-1} + \cdots + c_1 x + c_0 \tag{2.5}$$

and $b_i,\ c_i \in GF(2)$, for $i = 0,\ 1, \ldots, m-1$. Then (2.3) can be expressed as another form as follows:

$$c(x) = (\sum_{i=0}^{m-1} b_i a(x) x^i) \text{mod } f(x) \tag{2.6}$$

Then, (2.6) can be changed into

$$c(x) = \sum_{i=0}^{m-1} b_i (a(x) x^i \text{mod } f(x)) \tag{2.7}$$

Then define

$$a(x)^0 = a(x), \text{ and } a(x)^j = (a(x) x^j \text{mod } f(x)) \tag{2.8}$$

Then

$$a(x)^{j+1} = a(x)^i x \text{mod } f(x) \tag{2.9}$$

Define that

$$a(x)^j = \sum_{i=0}^{m-1} a_i^j x^i \tag{2.10}$$

Since x is the root of $f(x)$, then we can have

$$x^m = f_{m-1} x^{m-1} + \cdots + f_1 x + f_0 \tag{2.11}$$

Substituting (2.11) into (2.9)

$$a(x)^{j+1} = a_{m-1}^{j+1} x^{m-1} + \cdots + a_1^{j+1} x + a_0^{j+1} \tag{2.12}$$

where

$$\begin{aligned} a_0^{j+1} &= a_{m-1}^j \\ a_i^{j+1} &= a_{i-1}^j + a_{m-1}^j f_i, \text{for } 1 \le i \le m-1 \end{aligned} \tag{2.13}$$

The algorithm above presents the polynomial basis multiplication over $GF(2^m)$. It is worth mentioning that the choice of the irreducible polynomial $f(x)$ may reduce the computation complexity of modular reduction. Usually, the sparse irreducible polynomials having fewer nonzero terms are preferred for efficient realization of multiplication over $GF(2^m)$. In the following section, we will give some important irreducible polynomials which are widely used in modern cryptographic systems.

2.2.1 Important irreducible polynomials

As mentioned above, the choice of the irreducible polynomial $f(x)$ may ease the multiplication operations over $GF(2^m)$. Among all the important irreducible polynomials, usually trinomials, pentanomials, equally spaced polynomials, and AOPs are selected for real applications [1]-[9]. In this report, however, we only focus on trinomials, pentanomials and AOPs with several reasons as follows:

1 Trinomials are the shortest polynomials existed, and the multiplications are easy to be performed.

2 The National Institute of Standards and Technology (NIST) has recommended five binary finite fields for ECC implementation, and two of those are generated by the trinomials, $f(x) = x^{233} + x^{73} + 1$ and $f(x) = x^{409} + x^{87} + 1$.

3 The NIST also suggested three pentanomials for cryptographic engineering.

4 One of the pentanomial based multiplier suggested by NIST has the smallest size for cryptographic systems.

5 Multipliers for the AOP-based binary fields are simple and regular, very suitable for VLSI implementation.

6 AOP-based multiplication architectures can be used as a kernel circuit for field exponentiation, inversion, and division architectures.

7 AOP-based multipliers can be used for the nearly AOP (NAOP) which could be used for efficient realization of ECC systems.

2.2.2 AOPs

An AOP is a polynomial expressed as [1]-[5]

$$f(x) = x^m + x^{m-1} + \cdots + x + 1 \tag{2.14}$$

where all the coefficients are "1". For an AOP, if $m+1$ is a prime number and 2 is a primitive modulo $m + 1$, this AOP is called as irreducible AOP. For example, for the values of $m \in \{2,\ 4,\ 10,\ 12,\ 18, 28,\ 36,\ 52,\ 58,\ 60,\ 66,\ 82,\ 100,\ 106,\ 130,\ 138,\ 148,\ 162,\ 172,\ 178, \ldots\}$, the AOP is reducible.

AOP has an important property, for an AOP $f(x)$, we can have $x^m = 1 + x + \ldots + x^{m-1}$, and therefore we have $x^m + 1 = 1$. This property can be used to reduce the complexity of arithmetic operations.

Recently, an efficient class of polynomial [34], namely the NAOP is introduced for efficient cryptographic system realization. In practical implementation, AOP can be used to represent the NAOP to reduce the computation complexity, i.e., some coefficients of AOP can be replaced by "0" to represent the NAOP.

2.2.3 Trinomials

A trinomial over $GF(2^m)$ can be expressed as[1]-[5]

$$f(x) = x^m + x^k + 1 \tag{2.15}$$

where $1 < k < m$. The trinomial has only three nonzero coefficients and there is no other irreducible polynomial that has fewer nonzero coefficients than the trinomial. Irreducible trinomial basis multiplication over $GF(2^m)$ has drawn significant attention because it can reduce the complexity of multiplication. Moreover, there are abundant irreducible trinomials existed for every degree m.

2.2.4 Pentanomials

A pentanomial over $GF(2^m)$ can be expressed as[1]-[5]

$$f(x) = x^m + x^{k3} + x^{k2} + x^{k1} + 1 \tag{2.16}$$

where $1 < k1 < k2 < k3 < m$. A polynomial with five nonzero coefficients, irreducible pentanomials have drawn significant attention also because using them can reduce the complexity of finite field arithmetic in $GF(2^m)$. Furthermore, it was proved in that there exists either an irreducible trinomial or pentanomial of degree $m \in [2, 10,000]$, therefore an irreducible pentanomial can be used whenever an irreducible trinomial of degree m does not exist.

2.2.5 Existing works about polynomial basis multiplication over $GF(2^m)$

The multiplication over $GF(2^m)$ can be easily implemented on a software platform. But for a number of practical applications, like the credit card, it is unpractical to embed a general purpose machine in a card. Moreover, the software implementations usually do not meet the speed requirement of some time critical systems. Most of the real time cryptographic systems, therefore, need hardware implementations. The hardware implementation usually has highly cost in terms of area and time complexity. Therefore, the designing of a high-speed hardware structure with less area requirement has become much more demanding.

A number of works have been done on efficient realization of multiplication over $GF(2^m)$ based on irreducible polynomials [37]-[49]. These works can be classified into three major types, in terms of their input-output structuring: (i) serial-in serial-out structures, (ii) parallel-in parallel-out structures and (iii) serial/parallel structures. Serial-in serial-out structure usually has low area requirement but its throughput is low and usually it is not suitable for real applications. The parallel-in parallel-out structure, however, has a high throughput but suffers large area complexity. To tradeoff the area-time complexity, serial/parallel is introduced by some researches to achieve an optimal balance. These works can also be classified into two types, in terms of the design style, the systolic designs and the non-systolic designs.The non-systolic designs may have low latency, but yield low throughput, while the systolic designs feature modularity, regularity and local interconnections, which are important properties for VLSI design [17]-[18], and all the PEs in a systolic array are fully pipelined to produce very high throughput rate [19]-[20]. Along with the development of semiconductor technology, more and more transistors can be embedded in one single chip. On the other side, more and more modern cryptographic systems require high-speed processing abilities. Therefore, in this thesis, we only focus on the parallel-in parallel-out systolic structures for multiplication over $GF(2^m)$.

2.2.6 Existing works on AOPs

For non-systolic realizations: In one early paper [44], Fenn *et al*. has presented a serial-in serial-out structure for AOP-based multiplication. An efficient bit-parallel multiplier defined

by AOP has been shown in [45]. A serial-in serial-out multiplier defined by AOP has been suggested in [46]. Very recently, Meher *et al.* has suggested a serial/parallel structure for AOP-based multiplication [47].

For systolic realizations: In an early paper [48], a bit-parallel systolic design of multipliers for the field defined by AOP has been suggested by Lee *et al.* Another efficient bit-parallel systolic design is presented in [49]. In a recent paper [50], a low-complexity bit-parallel systolic Montgomery multiplier based on AOP has been suggested. Very recently [51], an efficient systolic Montgomery multiplier based on AOP is presented.

2.2.7 Existing works on trinomials

In an early paper [53], a bit-parallel systolic design for multiplication over $GF(2^m)$ have been suggested by Yeh *et al.*. Several other works have been suggested for efficient realization of finite field multiplication over $GF(2^m)$ [38]-[40]. In [36], the authors propose a systolic structure for polynomial based multiplication. In a recently reported paper [37], Meher has suggested systolic and super-systolic structures for multiplication over $GF(2^m)$.

2.2.8 Existing works on pentanomials

There are only a small number of papers specifically focusses on pentanomials [61]. And Because pentanomial is more complicate than the other polynomials, it is rare to find a design with low-latency high-throughput abilities. Structures about penatanomials proposed in this thesis are all the brand new designs.

2.2.9 Research direction

NIST has recommended five polynomials for ECC implementation, two of them are trinomials, the others are pentanomials [8] (unfortunately, AOP is not selected here). Trinomial based multiplier is simpler than pentanomial based multiplier, and hence is thoroughly studies in past years because of its characteristics like simplicity and regularity of the multiplication. On the other hand, low-latency high-throughput bit-level and digit-level multipliers based

on pentanomials are rarely reported due to their complicated multiplications, though three of them are recommended by NIST for real applications.

Keeping this in view, in this thesis, we will only focus on pentanomial based multipliers. Bit-parallel systolic multipliers are preferred for high-speed applications, but to lower the latency would be a real challenge. To obtain balance between area and time complexities, digit-serial multipliers would be a good choice. Overall, area-time efficient bit-parallel and digit-serial multipliers based on pentanomial will be the current/future research direction.

2.3 NORMAL BASIS MULTIPLICATION OVER $GF(2^M)$

Define an element $\beta \in GF(2^m)$, then $N = [\beta^{2^0},\ \beta^{2^1}, \ldots,\ \beta^{2^{m-1}}]$ is called a *Normal Basis* of $GF(2^m)$ over $GF(2)$ ($\beta^{2^0},\ \beta^{2^1}, \ldots,\ \beta^{2^{m-1}}$ are linearly independent). We can also say that β generates the normal basis N, or β is one of the normal elements of $GF(2^m)$ over $GF(2)$. It is already proved that for all positive integers m there exists a normal basis in the field $GF(2^m)$ over $GF(2)$ [1]-[5]. Based on the above definition, any element $A \in GF(2^m)$ can be expressed as (using a normal basis):

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = a_0\beta + a_1\beta + \cdots + a_{m-1}\beta^{2^{m-1}} \tag{2.17}$$

where $a_i \in GF(2)$ and $0 \leq i \leq m-1$ (a_i can be called the ith coordinates of A). For simplicity of discussion, A can be expressed as $A = (a_0,\ a_1,\ \ldots, a_{m-1})$ for short.

Squaring operation, carried out by only cyclic right shift, is the simplest and easiest operation in normal basis arithmetic. And hence there is nearly no cost in hardware realization, which automatically makes normal basis a preferred choice of representation in systems based on hardware implementation. On the other side, normal basis multiplication is not that simple as squaring, even much more complicated. Lots of efforts have been made to reduce the complexity of normal basis multiplication. Hardware realization of two two finite field elements represented in a normal basis was first described by Massey and Omura [69] (people usually call normal basis multiplication after their names as Massey-Omura

multipliers). Nevertheless, normal basis multipliers offer high complexity in hardware when compare to polynomial basis multipliers [70]-[87].

There are some efforts have been made to reduce the complexity of normal basis multiplier. Mullin *et al.* proposed a serial of normal bases, called ONB, which has lower complexity in hardware realization. There are two special types of normal bases, i.e., type-I and type-II ONB. RB is a variance of type-I ONB, while RNB is a reordered form of type-II ONB [68]. The use of the RB and RNB can reduce significantly the complexity of various arithmetic operations.

2.3.1 Some properties

Any two elements and $\beta \in GF(2^m)$, we have $(\alpha+\beta)^2 = \alpha^2+\beta^2$. Furthermore, for any element $\alpha \in GF(2^m)$, we can have $\alpha^{2^m} = \alpha$. Therefore, we can have $1 = \beta + \beta^2 + \beta^4 + \cdots + \beta^{2^{m-1}}$ for any element $\beta \in GF(2^m)$. We can say that for (1, 1, 1, ..., 1) is the normal basis representation of 1 [1]-[5].

Define $N = [\beta^{2^0}, \beta^{2^1}, \ldots, \beta^{2^{m-1}}]$ as a normal basis of $GF(2^m)$ over $GF(2)$ ($\beta^{2^0}, \beta^{2^1}, \ldots, \beta^{2^{m-1}}$ are linearly independent). While one polynomial has roots are linearly independent over $GF(2)$ in binary field is called an irreducible N-polynomial. Moreover, it has already been proven that the elements of a normal basis are exactly the same roots of an N-polynomial. Therefore, we can describe a normal basis as another N-polynomial.

For practical applications, the normal basis arithmetic operation should have a complexity as low as polynomial. Given an integer m and the field $GF(2)$, generate a normal basis of $GF(2^m)$ over finite field, or equivalently, construct an N-polynomial of degree m, is quite important.

2.3.2 Squaring

Define $N = [\beta^{2^0}, \beta^{2^1}, \ldots, \beta^{2^{m-1}}]$ is a normal basis of $GF(2^m)$ over finite field. For any A and $B \in GF(2^m)$, we can have $(A+B)^2 = A^2 + B^2$ since $2AB = 0$. From Fermats theorem, we can have $A^{2^{m-1}} = 1$ and $A^{2^m} = A$ [1]-[5]. Thus, we can have the following equations to depict the squaring process (Normal basis squaring operation is widely used in many

applications due to its efficient realization, i.e., it involves nearly free hardware cost since no extra operation is involved except cyclic shift):

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = a_0\beta + a_1\beta + \cdots + a_{m-1}\beta^{2^{m-1}} \tag{2.18}$$

then we can have

$$\begin{aligned} A^2 &= a_0\beta^{2^1} + a_1\beta^{2^2} + \cdots + a_{m-1}\beta^{2^m} \\ &= a_{m-1}\beta^{2^0} + a_0\beta^{2^1} + \cdots + a_{m-2}\beta^{2^{m-1}} \end{aligned} \tag{2.19}$$

Therefore, we can express $A = (a_0,\ a_1,\ \ldots,\ a_{m-1})$, $A^2 = (a^{m-1},\ a_0,\ \ldots,\ a^{m-2})$. From the above expression, we can see that squaring is executed by a simple cyclic right shift. And therefore there is almost no hardware involved in this operation.

2.3.3 Normal basis multiplication and its example

In this subsection, the work originally presented by Massey and Omura is briefly reviewed. Let $[\beta^{2^0},\ \beta^{2^1}, \ldots,\ \beta^{2^{m-1}}]$ be a normal basis of $GF(2^m)$ over finite field. And we define two elements A and B represented in the normal basis as $A = \sum_{i=0}^{m-1} a_i\beta^{2^i}$ and $B = \sum_{i=0}^{m-1} b_i\beta^{2^i}$, respectively. Represent A and B in vector notation by $A = (a_0,\ a_1,\ \ldots,\ a_{m-1})$ and $B = (b_0,\ b_1,\ \ldots,\ b_{m-1})$, respectively. Then the product C can be expressed as $C = AB = (c_0,\ c_1,\ \ldots,\ c_{m-1})$ in vector form [1]-[5].

Here we give an example to explain the normal basis multiplication [2].

Let $f(x) = x^4 + x^3 + 1$ be the generating irreducible polynomial for $GF(2^4)$ and α is the root of $f(x)$. Any two elements A and B in $GF(2^4)$ can be represented as $A = a_0\alpha + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^8$ and $B = b_0\alpha + b_1\alpha^2 + b_2\alpha^4 + b_3\alpha^8$. Therefore, the product $C = AB$ can be expressed as follows:

$$\begin{aligned} C = AB &= (a_0\alpha + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^8)(b_0\alpha + b_1\alpha^2 + b_2\alpha^4 + b_3\alpha^8) \\ &= c_0\alpha + c_1\alpha^2 + c_2\alpha^4 + c_3\alpha^8 \\ &= \alpha^{12}(a_2b_3 + a_3b_2) + \alpha^{10}(a_1b_3 + a_3b_1) + \alpha^9(a_3b_0 + a_0b_3) \\ &+\alpha^8(a_2b_2) + \alpha^6(a_2b_1 + a_1b_2) + \alpha^5(a_2b_0 + a_0b_2) \\ &+\alpha^4(a_1b_1) + \alpha^3(a_0b_1 + a_1b_0) + \alpha^2(a_0b_0) + \alpha(a_3b_3) \end{aligned} \tag{2.20}$$

and then based on the above equations, we can have following steps to show a detailed multiplication process:

$$\begin{aligned} c_3 &= a_2b_2 + a_3b_2 + a_2b_3 + a_3b_1 + a_1b_3 + a_3b_0 + a_0b_3 + a_1b_0 + a_0b_1 \\ c_2 &= a_1b_1 + a_2b_1 + a_1b_2 + a_2b_0 + a_0b_2 + a_2b_3 + a_3b_2 + a_0b_3 + a_3b_0 \\ c_1 &= a_0b_0 + a_1b_0 + a_0b_1 + a_1b_3 + a_3b_1 + a_1b_2 + a_2b_1 + a_3b_2 + a_2b_3 \\ c_0 &= a_3b_3 + a_0b_3 + a_3b_0 + a_0b_2 + a_2b_0 + a_0b_1 + a_1b_0 + a_2b_1 + a_1b_2 \end{aligned} \tag{2.21}$$

while the other normal basis multiplications have similar operations as (2.20) and (2.21).

2.3.4 Existing works on RB (Type-I ONB)

Several digit-level serial/parallel structures for RB multiplier over $GF(2^m)$ have been reported in the last years [68], [70]-[73] after its introduction by Wu *et al*. [68]. An efficient serial/parallel multiplier using redundant representation has been presented in [70]. A bit-serial word-parallel (BSWP) architecture for RB multiplier has been reported by Namin *et. al* [71]. Several other RB multipliers also have been developed by the same authors in [72]-[73] for reducing the complexity of implementation and for high-speed realization. We find that the hardware utilization efficiency and throughput of existing structures of [70]-[73] can be improved by efficient design of algorithm and architecture.

2.3.5 Existing works on RNB (Type-II ONB)

There are two special classes of normal basis for which the complexity of multiplication can be minimized, namely the ONB type-I and -II. ONB Type-II has been selected as potential candidate for various cryptographic system designs. RNB is a version of ONB type II which has been proposed in [67] for efficient multiplication implementation. Later, efficient multipliers are suggested in [68] based on this basis. Very recently, two high-speed architectures for multiplication using RNB are proposed in [80].

2.3.6 Research direction

Normal basis multipliers usually are complex and once they are not preferred for cryptographic systems because of their high area-complexity of multiplication operation. But through years' derivation, some reordered versions, like RB and RNB, merged that the normal basis multipliers have great potential in cryptographic systems especially because of its free squaring operations. On the other hand, because of their high-complexity in hardware implementation, resource reusable technique, i.e., hybrid architecture, will be much more demanding in the future.

Based on the above consideration, in this thesis, we will not only focus on normal basis multipliers with low-complexity and high-throughput capabilities, but also will develop a hybrid architecture with various throughput rate choices. Overall, area-time efficient bit-parallel and digit-serial single and hybrid multipliers based on normal basis will be the current/future research direction.

3.0 LOW LATENCY SYSTOLIC MONTGOMERY MULTIPLIER FOR FINITE FIELD $GF(2^M)$ BASED ON PENTANOMIALS

In this chapter, we present a low latency systolic Montgomery multiplier over $GF(2^m)$ based on irreducible pentanomials. We have presented an efficient algorithm to decompose the multiplication into a number of independent units for parallel processing. Besides, we have introduced a novel so-called PCA technique to further reduce the latency. We have also proposed a NMR operation based on the PCA technique, which is more suitable for deriving low latency multiplier compared to traditional modular reduction schemes in existing multipliers. Moreover, by suitable cut-set retiming, we have derived here a low latency bit-level-pipelined systolic design for field multiplication based on our proposed algorithm. The proposed design has the same critical-path as the corresponding existing design, but offers at least one-fourth of the latency of the other. For the pentanomial suggested by NIST, the proposed design offers significant reduction of area-delay product over the recently reported design. The proposed design is simple and modular, and therefore suitable for VLSI implementation.

3.1 INTRODUCTION

Cryptographic applications in ECC and error control coding systems require field operations over $GF(2^m)$ [1]-[5]. The implementation of multiplication over $GF(2^m)$ greatly impacts the overall system performance. Accordingly, many efforts have been made for efficient realization of multiplication over $GF(2^m)$ [3]-[14]. Out of the three bases, i.e., dual basis, normal basis, and polynomial basis, the polynomial basis designs have gained much more

popularity compared with the multipliers due to its several advantages over the other two bases [5-6]. The pentanomial based Galois field is widely used in public-key cryptography systems, since the NIST has recommended three pentanomials for ECC application [8].

All the existing designs for multiplication over $GF(2^m)$ based on polynomials could be categorized as non-systolic and systolic designs [8]-[15]. Both the non-systolic and systolic designs have advantages. The systolic designs have efficiency in area-time complexity, being supported by their features such as modularity and regularity of the structures [16]-[17], and usually, the systolic structures have high throughput rate.

The structures for multiplication over $GF(2^m)$ based on irreducible trinomials and general polynomials have been extensively explored [37]-[40]. There are, however, only a few systolic realizations of pentanomial-based multiplier. In a recent report [25], Meher has presented an efficient systolic structure for multiplication over $GF(2^m)$ based on irreducible pentanomial. The design involves significantly less area-time complexity compared with other designs.

The systolic structure for field multiplication in [25] has a latency of m cycles. In this section, we have extended further to obtain a lower latency systolic structure for multiplication over $GF(2^m)$ based on irreducible pentanomial. First of all, we have shown an efficient algorithm for Montgomery multiplication over $GF(2^m)$ based on irreducible pentanomial. It is shown that the multiplication can be decomposed into a number of independent components, which could be processed in parallel without changing the critical-path. Furthermore, we have introduced a novel PCA technique such that the latency of a multiplier can be reduced further. Accordingly, a NMR operation is introduced based on the PCA technique. The proposed structure achieves significantly less time complexity than the corresponding existing structure.

3.2 ALGORITHM

In this section, we firstly present an efficient Montgomery algorithm, which can reduce the latency of the multiplier followed by the proposed PCA technique. Some examples and

extensions of the proposed PCA technique are also given as well as the detailed algorithm steps and processes.

3.2.1 Montgomery multiplication

Let $f(x)$ be an irreducible polynomial over $GF(2)$ as

$$f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + 1 \tag{3.1}$$

where $f_j \in GF(2) = \{0, 1\}.\{1, x, , x^{m-1}\}$ is the polynomial basis in $GF(2^m)$, such that we can have the Montgomery multiplication algorithm as [2]

$$C = A \cdot B \cdot r^{-1} \mathrm{mod} f(x) \tag{3.2}$$

where

$$\begin{aligned} A &= a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 \\ B &= b_{m-1}x^{m-1} + \cdots + b_1 x + b_0 \\ C &= c_{m-1}x^{m-1} + \cdots + c_1 x + c_0 \end{aligned} \tag{3.3}$$

and $a_j, b_j, c_j \in GF(2)$, for $j = 0, 1, , m-1$. r is Montogomery factor that satisfies $gcd(r, f(x)) = 1$, where gcd means the greatest common divisor. In [78], $r = x^{m-1}$.

In practical applications, m is an odd number, such as the pentanomial suggested by NIST, e.g., $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. In that case we can define $r = x^u = x^{(m-1)/2}$ as the Montgomery factor, and then (3.2) can be expressed as

$$C = \sum_{i=0}^{m-1} b_i (A \cdot x^i \cdot x^{-u} \mathrm{mod} f(x)) = C^- + C^+ \tag{3.4}$$

where

$$\begin{aligned} &\textstyle\sum_{i=0}^{u-1} b_i \cdot A \cdot x^{i-u} \mathrm{mod} f(x)) = C^- \\ &\textstyle\sum_{i=u}^{m-1} b_i \cdot A \cdot x^{i-u} \mathrm{mod} f(x)) = C^+ \end{aligned} \tag{3.5}$$

Let us define

$$A^{(0)} = A \tag{3.6}$$

and

$$A^{(1)} = A \cdot x^{-1} \mathrm{mod} f(x) A^{(2)} = A \cdot x^{-2} \mathrm{mod} f(x) \ldots A^{(u)} = A \cdot x^{-u} \mathrm{mod} f(x) \tag{3.7}$$

Such that based on the above discussions, we can have the following steps to give a detailed description of proposed technique:

$$A^{(i+1)} = (A^{(i)} \cdot x^{-1} \mathrm{mod} f(x)) \tag{3.8}$$

For simplicity of discussion, (3.5) can be rewritten as

$$C^{-} = \sum_{i=1}^{u} X_i \tag{3.9}$$

where $X_i = b_{u-i} A(i)$.

Therefore, we can have

$$A^{(i+1)} = a_0^{(i)} x^{-1} + a_1^{(i)} + \cdots + a_{m-1}^{(i)} x^{m-2} \tag{3.10}$$

where

$$A^{(i)} = a_0^{(i)} + a_1^{(i)} x + \cdots + a_{m-1}^{(i)} x^{m-1} \tag{3.11}$$

Since x is the root of $f(x)$, one can also have $x^m + f_{m-1} x^{m-1} + \cdots + f_1 x = 1$ and $x^{-1} = x^{m-1} + f_{m-1} x^{m-2} + \ + f_1$. After substituting x^{-1} into (3.10), we can find

$$A^{(i+1)} = a_0^{(i+1)} + a_1^{(i+1)} + \cdots + a_{m-1}^{(i+1)} x^{m-1} \tag{3.12}$$

where

$$\begin{aligned} a_{m-1}^{(i+1)} &= a_0^{(i)} \\ a_j^{(i+1)} &= a_{j+1}^{(i)} \bigoplus f_j \cdot a_0^{(i)} \end{aligned} \tag{3.13}$$

for $j = 1, 2, \ldots, m-1$.

For the C^{+} of (3.5), similarly, we can have

$$C^{+} = \sum_{i=u}^{m-1} Y_i \tag{3.14}$$

where $Y_i = b_i A'^{(i)}$, for $A'^{(i)} = A$ and $A'^{(i)} = (A \cdot x^{i-u} \mathrm{mod} f(x))$. Such that $A'^{(i+1)}$ can be obtained from $A'^{(i)}$ recursively as $A'^{(i+1)} = (A'^{(i)} \cdot x \ \mathrm{mod} f(x))$.

We can also have

$$A'^{(i+1)} = [a_0'^{(i)} x + a_1'^{(i)} x^2 + \cdots + a_{m-1}'^{(i)} x^m] \ \mathrm{mod} f(x) \tag{3.15}$$

$$A'^{(i)} = a_0'^{(i)} + a_1'^{(i)}x + \cdots + a_{m-1}'^{(i)}x^{m-1} \tag{3.16}$$

Substituting x^m into (3.15), we find

$$A'^{(i+1)} = a_0'^{(i+1)} + a_1'^{(i+1)}x + \cdots + a_{m-1}'^{(i+1)}x^{m-1} \tag{3.17}$$

where

$$\begin{aligned} a_0'^{(i+1)} &= a_{m-1}'^{(i)} \\ a_j'^{(i+1)} &= a_{j-1}'^{(i)} \bigoplus f_j \cdot a_{m-1}'^{(i)} \end{aligned} \tag{3.18}$$

for $j = 1, 2, m-1$. Let us define C^+ of (3.5) as the forward block, and C^- as the inverse block. Similarly, we define the operation of (3.13) as inverse modular reduction (IMR) operation, then the operation of (3.18) can be defined as forward MR (FMR) operation. Each of the IMR and FMR operations requires a duration of T_X, where T_X is propagation delay of XOR gate.

One can define an irreducible pentanomial of degree m as $f(x) = x^m + x^{k1} + x^{k2} + x^{k3} + 1$, for $1 \leq k3 < k2 < k1 \leq m-1$, and comparing pentanomial with (3.1), one can have

$$f_j = \begin{cases} 1 \ \text{ for } j = k1, k2, k3 \\ 0 \ \text{ for } 1 \leq j \leq m-1 \text{ and } j \neq k1, k2, k3 \end{cases} \tag{3.19}$$

Then, we can substitute (3.19) into (3.13) and (3.15) to obtain the details of the IMR and FMR operations. It is possible to extend IMR and FMR operations to derive the reduced forms $A'^{(i+l)}$ ($A^{(i+l)}$) concurrently from $A'^{(i)}$ ($A^{(i)}$) for reducing the degree by l for $l = \min\{m-k1, k1-k2, k2-k3\}$, as follows

$$a_j'^{(i+l)} = \begin{cases} a_{m-l+j}'^{(i)} \ \text{ for } 0 \leq j \leq l-1 \\ a_{j-l}'^{(i)} \bigoplus a_{m-l+j-kl}'^{(i)} \ \text{ for } k1 \leq j \leq k1+l-1 \\ a_{j-l}'^{(i)} \bigoplus a_{m-l+j-k2}'^{(i)} \ \text{ for } k2 \leq j \leq k2+l-1 \\ a_{j-l}'^{(i)} \bigoplus a_{m-l+j-k3}'^{(i)} \ \text{ for } k3 \leq j \leq k3+l-1 \\ a_{j-1}'^{(i)} \ \text{ otherwise} \end{cases} \tag{3.20}$$

Note that the inverse block involves similar operations as the forward block, which is the same as in rest of the chapter.

Now let us consider $u = l \times P + r$, where l, P and r are integers, and $0 \leq P \leq l$ and $0 \leq r \leq l$, we can rewrite (3.7) in a form

$$\begin{aligned} & C^- = \textstyle\sum_{i=1}^{u} X_i = (X_1 + X_{l+1} + \cdots + X_{Pl+1}) \longrightarrow \text{1st unit} \\ & + \cdots + \\ & (X_r + X_{l+r} + \cdots + X_u) \longrightarrow r\text{th unit} \\ & + \cdots + \\ & (X_l + X_{2l} + \cdots + X_{Pl}) \longrightarrow l\text{th unit} \end{aligned} \tag{3.21}$$

Similarly, for $(m - u) = l \times Q + t$, where l, Q and t are integers, and $0 \leq Q \leq l$ and $0 \leq t \leq l$, we can express (3.11) as

$$\begin{aligned} & C^+ = \textstyle\sum_{i=u}^{m-1} Y_i = (Y_u + Y_{l+u} + \cdots + Y_{Ql+u}) \longrightarrow \text{1st unit} \\ & + \cdots + \\ & (Y_{u+t} + Y_{l+u+t} + \cdots + Y_{m-1}) \longrightarrow r\text{th unit} \\ & + \cdots + \\ & (Y_{u+L-1} + Y_{2l+u-1} + \cdots + Y_{u+Ql-1}) \longrightarrow l\text{th unit} \end{aligned} \tag{3.22}$$

Therefore, according to (3.21) and (3.22), the multiplier can be implemented by $2l$ parallel units having the same critical-path.

3.2.2 PCA technique

For pentanomials with l=1, e.g., $f(x) = x^{13} + x^4 + x^3 + x^2 + 1$, the derivation of $A'^{(i+l)}$ ($A^{(i+l)}$) concurrently from $A'^{(i)}$ ($A^{(i)}$) requires more logic-time as the degree of modular reduction"l"increases. Therefore, we introduce a novel PCA technique here to reduce this delay. For simplicity of discussion, we illustrate here only the forward block. For $f(x) = x^{13} + x^4 + x^3 + x^2 + 1$, from (3.13) we can find that

$$\begin{aligned} & C^+ = \textstyle\sum_{i=6}^{12} Y_i \\ & = (A'^{(6)}b_6 + A'^{(8)}b_8 + A'^{(10)}b_{10} + A'^{(12)}b_{12}) \longrightarrow \text{1st unit} \\ & (A'^{(7)}b_7 + A'^{(9)}b_9 + A'^{(11)}b_{11}) \longrightarrow \text{2nd unit} \end{aligned} \tag{3.23}$$

Define $M_1 = [A'^{(6)}\ A'^{(8)}\ A'^{(10)}\ A'^{(12)}]$ and $M_2 = [A'^{(7)}\ A'^{(9)}\ A'^{(11)}]$, and the detail of M_1 and M_2 is in Fig. 1(a) and (b), respectively.

$$M_1 = \begin{bmatrix}
a_0 & a_{11} & a_9 & a_7 \\
a_1 & a_{12} & a_{10} & a_8 \\
a_2 & a_0 + a_{11} & a_{11} + a_9 & a_9 + a_7 \\
a_3 & a_1 + a_{12} + a_{11} & a_{12} + a_{10} + a_9 & a_{10} + a_8 + a_7 \\
a_4 & a_2 + a_{12} + a_{11} & a_0 + a_{11} + a_{10} + a_9 & a_{11} + a_9 + a_8 + a_7 \\
a_5 & a_3 + a_{12} & a_1 + a_{12} + a_{11} + a_{10} & a_{12} + a_{10} + a_9 + a_8 \\
a_6 & a_4 & a_2 + a_{12} + a_{11} & a_0 + a_{11} + a_{10} + a_9 \\
a_7 & a_5 & a_3 + a_{12} & a_1 + a_{12} + a_{11} + a_{10} \\
a_8 & a_6 & a_4 & a_2 + a_{12} + a_{11} \\
a_9 & a_7 & a_5 & a_3 + a_{12} \\
a_{10} & a_8 & a_6 & a_4 \\
a_{11} & a_9 & a_7 & a_5 \\
a_{12} & a_{10} & a_8 & a_6
\end{bmatrix}$$

(a)

$$M_2 = \begin{bmatrix}
a_{12} & a_{10} & a_8 \\
a_0 & a_{11} & a_9 \\
a_1 + a_{12} & a_{12} + a_{10} & a_{10} + a_8 \\
a_2 + a_{12} & a_0 + a_{11} + a_{10} & a_{11} + a_9 + a_8 \\
a_3 + a_{12} & a_1 + a_{12} + a_{11} + a_{10} & a_{12} + a_{10} + a_9 + a_8 \\
a_4 & a_2 + a_{12} + a_{11} & a_0 + a_{11} + a_{10} + a_9 \\
a_5 & a_3 + a_{12} & a_1 + a_{12} + a_{11} + a_{10} \\
a_6 & a_4 & a_2 + a_{12} + a_{11} \\
a_7 & a_5 & a_3 + a_{12} \\
a_8 & a_6 & a_4 \\
a_9 & a_7 & a_5 \\
a_{10} & a_8 & a_6 \\
a_{11} & a_9 & a_7
\end{bmatrix}$$

(b)

Figure 1: Detailed expression of M_1 and M_2. (a) The detail of M_1. (b) The detail of M_2.

$$\begin{bmatrix}
\vdots \\
a_{12} + a_{10} \\
a_0 + a_{11} + a_{10} \\
a_1 + a_{12} + a_{11} + a_{10} \\
a_2 + a_{12} + a_{11} \\
a_3 + a_{12} \\
\vdots
\end{bmatrix} = \begin{bmatrix}
\vdots \\
(a_{12}) + (a_{10}) \\
(a_0) + (a_{11} + a_{10}) \\
(a_1 + a_{12}) + (a_{11} + a_{10}) \\
(a_2 + a_{12}) + (a_{11}) \\
(a_3) + (a_{12}) \\
\vdots
\end{bmatrix}$$

Figure 2: An example of bit-addition involved in M_2.

It is clear from Fig. 1 that if we want to derive the operand $A'^{(i+2)}$ directly from $A'^{(i)}$, it would involve a delay of $2T_X$. To reduce this duration, we introduce here a novel PCA technique. Let us first take M_2 as illustration. It is observed that some bits of $A'^{(9)}$ of M_2 can be expressed as addition of two bits, as shown in Fig. 2, e.g., $a_0+a_{11}+a_{10} = (a_0)+(a_{11}+a_{10})$. It is also observed that if we add an additional value $(a_{11} + a_{10})$, then we can obtain $A'^{(9)}$ directly from $A'^{(7)}$ in time T_X only, as shown in Fig. 3, while the other bits of $A'^{(7)}$) can

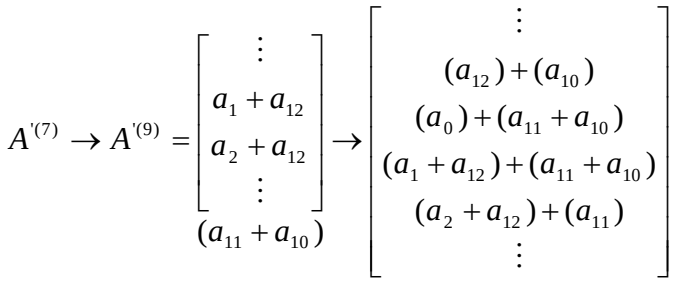

Figure 3: Process of derivation of $A'^{(9)}$ directly from $A'^{(7)}$.

be obtained through bits-shifting/ adding, e.g., $a_{12} + a_{10} = (a_{10}) + (a_{12})$. We can compute $(a_{11} + a_{10})$ concurrently during the time we obtain $A'^{(7)}$ from $A'^{(6)}$. Such pre-computed addition (PCA) therefore can be used to reduce the delay to obtain $A'^{(9)}$ directly from $A'^{(7)}$ from $2T_X$ to T_X. Similarly, we can obtain $A'^{(11)}$ directly from $A'^{(9}$ in time of T_X. The process can be expressed as Fig. 4(a), where the bits contained in the brackets are the PCA bits. Then, we can also have the process of Fig. 4(b).

Accordingly, the inverse block can have similar operations as those shown in Fig. 4. The PCA technique can be used in case of $l \geq 1$ such that (3.21) and (3.22) can be expressed further as follows:

$$\begin{aligned} C^- &= \textstyle\sum_{i=1}^{u} X_i \\ &= (X_1 + X_{l+2} + \cdots) \longrightarrow \text{1st unit} \\ &+ \cdots + \\ &(X_r + X_{l+r+1} + \cdots) \longrightarrow r\text{th unit} \\ &+ \cdots + \\ &(X_{l+1} + X_{2l+2} + \cdots) \longrightarrow (l+1)\text{th unit} \end{aligned} \tag{3.24}$$

and

$$\begin{aligned} &C^+ = \textstyle\sum_{i=u}^{m-1} Y_i = (Y_u + Y_{l+u+1} + \cdots) \longrightarrow \text{1st unit} \\ &+ \cdots + \\ &(Y_{u+t} + Y_{l+u+t+1} + \cdots) \longrightarrow r\text{th unit} \\ &+ \cdots + \\ &(Y_{u+L} + Y_{2l+u+1} + \cdots) \longrightarrow (l+1)\text{th unit} \end{aligned} \tag{3.25}$$

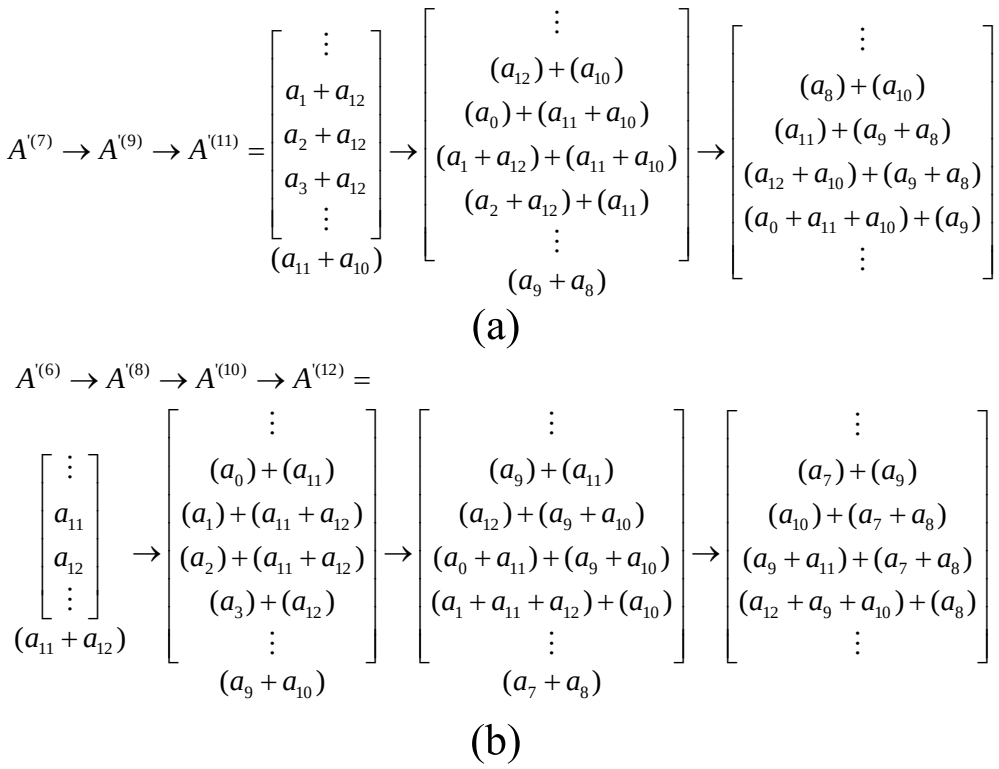

Figure 4: Process of PCA technique. (a) Process of derivation. (b) Process of derivation.

where each of the last terms in each unit depends on the pentanomial-used. It is clear that the multiplier can be realized by $(2l + 2)$ parallel units, and consequently the latency is reduced further compared to that of (3.24) and (3.25). The PCA technique allows the derivation of $A'^{(i+l+1)}$ and $A^{(i+l+1)}$ from $A'^{(i)}$ and $A^{(i)}$, respectively, to remain unchanged with number of degree "l"to be reduced.

Note that the operations of Fig. 4 are defined as novel MRs (NMR)s. The inverse block has similar operations. Note that the process of deriving $A'^{(11)}$ directly from $A'^{(9)}$ (see Fig. 4(a)) is also an NMR operation, though there are no additional bits contained in the operand $A'^{(11)}$. Similar operation can be seen in Fig. 4(b) as well.

The multiplication over $GF(2^m)$ based on pentanomials can be performed according to (3.24) and (3.25) as follows:

1 Use the proposed Montgomery algorithm to represent the multiplier into the two blocks, as shown in (3.5).

2 Represent each block of (3.5) in the form of (3.18) and (3.20), and use the proposed PCA technique to have (3.21) and (3.22).

3 Use the signal-flow graph (SFG) to represent (3.21) and (3.22), and derive the structure by suitable cut-set retiming [12].

3.3 PROPOSED STRUCTURES

Based on the proposed algorithm, we derive here the proposed structure of the multiplier. We have taken $f(x) = x^{13} + x^4 + x^3 + x^2 + 1$ as the irreducible pentanomial to illustrate the proposed low latency systolic structure (for $l = 1$). It can however easily be extended to other pentanomials.

From C^+ of (3.22), similarly, we can get C^- as

$$\begin{aligned} C^- &= (A'^{(1)}b_4 + A'^{(3)}b_2 + A'^{(5)}b_0) \longrightarrow \text{3rd unit} \\ &(A'^{(2)}b_5 + A'^{(4)}b_3 + A'^{(6)}b_1) \longrightarrow \text{4th unit} \end{aligned} \tag{3.26}$$

For simplicity of discussion, we have defined the two parallel units in (3.26) as the 3rd and 4th units, respectively, those are different from the definition in (3.21). Using the proposed PCA technique, one can derive $A'^{(i+2)}$ and $A^{(i+2)}$ from $A'^{(i)}$ and $A^{(i)}$, which involves a delay of only T_X. Both the forward and inverse blocks consist of three major steps, i.e., the NMR operation (Fig. 4), the bit-multiplication operation, and the bit-addition operation ((3.8) and (3.11)). For systolic implementation of multiplication over $GF(2^m)$, the operations of these steps are represented by the SFG of Fig. 5, where each section (within the dotted box, only two units are presented) corresponds to one of the units of (3.21). The 1st unit consists of 3 nodes R, 3 bit-multiplication nodes M(i) and 2 bit-addition nodes AD, while the 2nd unit has one more node R, one more node M(i) and one more node AD compared with the 1st unit. Besides, 2 bit-addition nodes AD are required for the final addition of these units. Functions of node R, node M(i) and node AD are depicted in Fig. 5(b), (c)

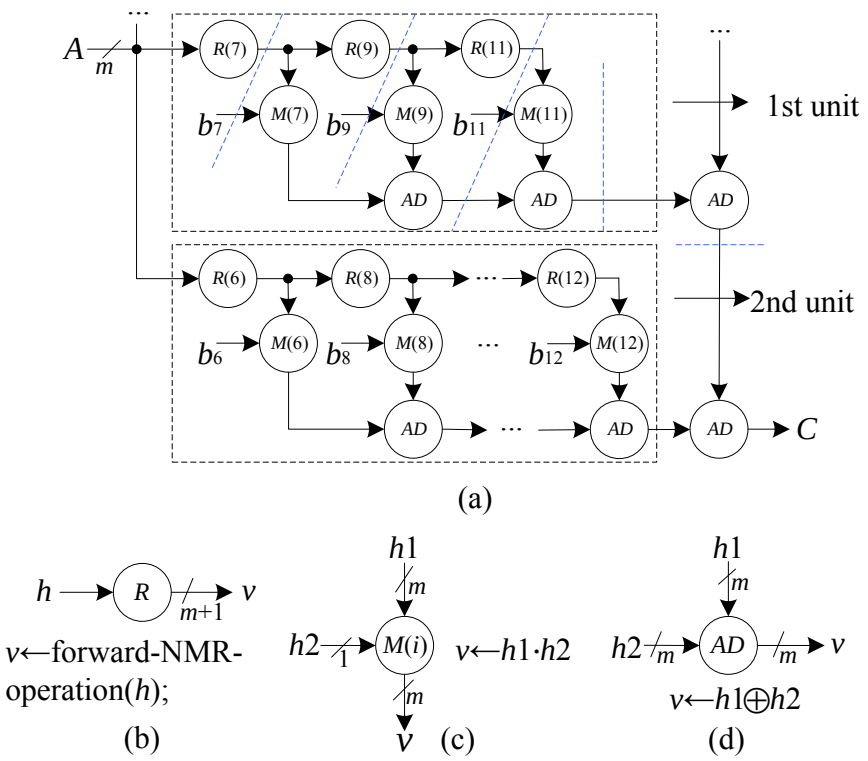

Figure 5: SFG for forward block of (3.21) for m=13. (a) SFG and cut-set retiming. (b) Functional description of node R. (c) Functional description of the ith bit-multiplication node M(i). (d) Functional description of the bit-addition node AD.

and (d), respectively. To reduce the critical-path further, a cut-set retiming (shown in Fig. 5(a)) is performed to introduce a delay between the NMR node and its corresponding bit-multiplication and bit-addition nodes to form the processing elements (PE), where we just show the 1st unit as an example for illustration.

The proposed systolic-like structure for field multiplication over $GF(2^m)$ based on pentanomial (for $m = 13$) is shown in Fig. 6(a). It consists of 4 systolic arrays, where each of the arrays corresponds to one of the units in (3.21) and (3.23). The first systolic array consists of 5 PEs, while each of the other arrays (2nd, 3rd, and 4th systolic arrays) consists of 4 PEs and a delay cell (the delay cell is required to meet the data dependence requirement). Although one can use a systolic adder array consisting of 4 addition cells (ACs) for the final

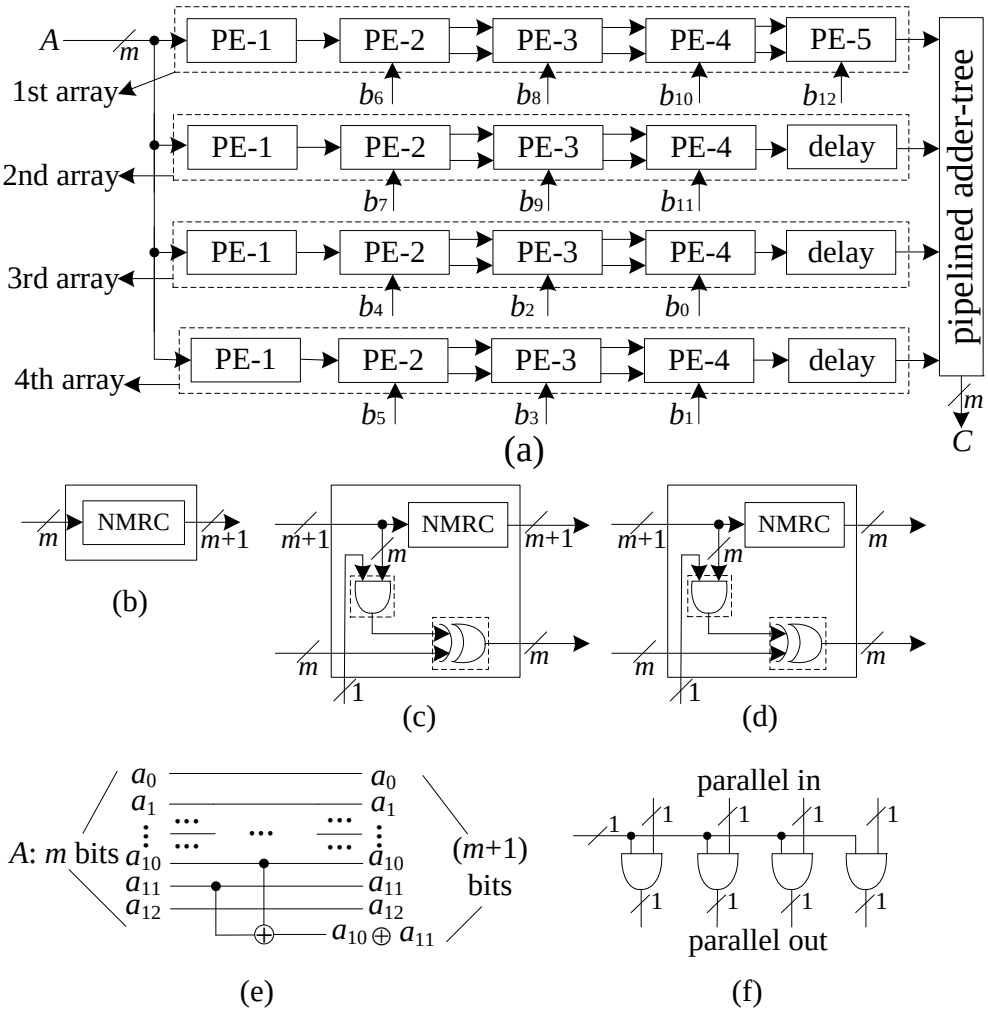

Figure 6: Proposed systolic-like structure for m=13. (a) Proposed structure. (b) Design of the PE-1. (c) Design of a regular PE. (d) Design of the second PE in the array (from right). (e) Structure of NMRC in the PE-1 of the 2nd array. (f) An example of structure of the AND cell.

addition of the 4 arrays, we use a pipelined adder-tree consisting of 3 ACs for a low latency implementation. The four arrays can function concurrently, such that after 5 cycles, the adder-tree receives its first input and yields its first output in 2 cycles.

The detailed design of the PEs of the proposed structure is shown in Fig. 6. PE-1 of the arrays contains only one NMR cell (NMRC), as shown in Fig. 6(b). The regular PE, as

Table 1: Hardware- and time-complexities of proposed and the existing structures for multiplication over $GF(2^m)$ based on pentanomials

design	NAND	AND	XOR	register	latency	CP
[25][1]	m^2	0	m^2+2m-1	$2m^2-2m$	m	T_N+T_X
[56]	0	$2m^2+2m$	m^2+m	$3.5m^2+3.5m$	$m+1$	T_A+2T_X
[57]	0	$2m^2$	$2m^2$	$7m^2$	$3m$	T_A+T_X
[58]	0	$2m^2$	$2m^2$	$3m^2$	$m+1$	T_A+T_X
[59]	0	m^2	$2m^2+m$	$2m^2$	m	T_A+2T_X
proposed[2]	m^2	0	$2m+2lm$	$2m^2-2m$	α^1	T_A+T_{XN}

[1]: There are extra m^2 number of INV gates, which are not listed here.

[2]: There are extra m^2 number of XNOR gates, which are not listed here.

α^1: $\alpha^1 = m/(2l+2)$

shown in Fig. 23(c) consists of one AND cell, one XOR cell and one NMRC. The AND cell and XOR cell correspond to the bit multiplication node and bit addition node of the SFG of Fig. 5, respectively. Each AND cell and XOR cell, respectively, consists of m number of AND gates and XOR gates working in parallel. Fig. 6(f) gives an example of the design, where m is assumed to be 4. Note that in PE-3 (PE-4 in the 1st array), as shown in Fig. 6(d), the output of NMRC consists of m bits. Moreover, we have also shown the design detail of the NMRC (PE-1 of the 2nd array) in Fig. 6(e). Note that the number of XOR gates in NMRC in the PEs depends on the generating irreducible pentanomial.

Since the delay of NMRC is only T_X, the duration of a PE amounts to $T_X + T_A = \max T_X$, $T_X + T_A$, where T_A is the delay of an AND gate. There is one additional XOR gate in the NMRC except the PE-3 in the array (PE-4 in the 1st array). In the previous work of [9], the author has used NAND gate to replace the AND gate in a PE, here we extend further to use XNOR gate to replace XOR gate such that the inverter (INV) in the PE can be saved.

3.4 HARDWARE AND TIME COMPLEXITY

Here, we discuss the estimation of hardware and time complexity of the proposed structure and compare that with that of the existing designs. For any pentanomials, the proposed structure requires $(2l + 2)$ parallel arrays. If we define $P = Q = (2l + 2)$, then each array requires nearly $(m/(2l + 2) + 1)$ XOR gates and $(m^2/(2l + 2))$ NXOR gates. The pipelined adder-tree requires $(2lm + m)$ XOR gates. Besides, m^2 NAND gates and nearly $(2m^2 - 2m - 2lm - 2l - 2)$ bit-registers are used in the structure. It is noted that the actual gate-counts may vary with different pentanomials, though this variation is minor. The gate-counts, register-counts, latency in cycles and critical-path of the proposed structure and the existing systolic structures of [25] and [56]-[59] (general polynomial (GP)) are listed in Table 1, for pentanomial-based systolic multiplier. For l=1, the proposed design has a latency of (m/4+3) cycles, so that for the NIST recommended pentanomials 163, 283 and 571, it involves nearly one-fourth of the latency of the structure of [25]. For the example of proposed structure described in Section III, we have latency = 7 cycles (for m=13), which is nearly

Table 2: Comparison of area and time complexities

design	area (nm^2)	CP (ns)	power	ADP	PDP	latency
[25]	5693	0.13	1.24	740	0.16	1.69
proposed	4782	0.13	1.04	622	0.14	0.91

half of the other. It is clear that the proposed design outperforms the existing designs. Although the proposed design has nearly the same gate-counts as [25], the proposed one requires shorter latency. Compared with [56] and [57], the proposed design not only has less area-complexity, but also has shorter latency. Designs of [58] and [59] involve either larger area or larger time-complexity when compared with the proposed one.

The proposed design (Fig. 6) has been coded in VHDL and synthesized by Synopsys Design Compiler using TSMC 90nm library [43] for $f(x) = x^{13}+x^4+x^3+x^2+1$ along with the best of the existing designs of [25]. The critical-path (CP), area and power consumption (at 100MHz frequency) thus obtained are listed in Table 2. The proposed design has nearly 16% less area-delay product (ADP), 12.5% lower power-delay product (PDP) and 46% shorter latency compared to the existing one.

3.5 CONCLUSTION

In this chapter, we have presented a novel PCA technique and modular reduction scheme for Montgomery multiplication over $GF(2^m)$ based on irreducible pentanomials. To illustrate the efficiency of the proposed approach, we have designed the multiplier for the irreducible pentanomial $f(x) = x^{13} + x^4 + x^3 + x^2 + 1$, for simplicity of discussion. We have decomposed the Montgomery multiplication into two concurrent blocks and we have derived a lower-latency multiplier using the proposed modular reduction scheme using PCA. The proposed design involves significantly less area-delay and power-delay complexities than the newly

reported multiplier for irreducible pentanomial, with nearly one-fourth of the latency of the other, for the NIST recommended pentanomials.

4.0 LOW-LATENCY HIGH-THROUGHPUT SYSTOLIC MULTIPLIERS OVER $GF(2^M)$ FOR NIST PENTANOMIALS

In this chapter, we present three pairs of low-latency and high-throughput bit-parallel and digit-serial systolic multipliers specifically based on NIST pentanomials. Novel decomposition-technique has been proposed that the multiplier is decomposed into several parallel processing arrays to obtain a bit-parallel systolic structure (BP-I) with a critical-path of $2T_X$. These parallel arrays are then projected along vertical direction to obtain a digit-serial structure (DS-I) with the same critical-path. To increase the throughput rate, another pair of bit-parallel (BP-II) and digit-serial (DS-II) structures are then presented based on a novel modular reduction operation, where the critical-paths are reduced to $T_A + T_X$. Identical data sharing between PEs has been proposed to reduce area-complexity of BP-I and BP-II further. Finally, we have proposed KA-based bit-parallel (BP-III) and digit-serial (DS-III) multipliers to enhance the throughput rate further. This is the first report on low-latency systolic multipliers based on NIST pentanomials without any restriction on latency choice.

4.1 INTRODUCTION

Finite field multipliers are widely used in various cryptographic systems such as ECC and error control coding [1-2]. A good multiplication design depends on the choice of a basis. Basically, there are three bases of representation, e.g., dual basis, normal basis, and polynomial basis, have been widely used in practical application [3]-[5]. Among the three basis multipliers, the polynomial-based designs have gained much more popularity compared with the multipliers based on the other two representations [6]-[7]. The pentanomial is one of

the most important polynomials which have been widely used in public-key cryptography systems. And the NIST [8] has also recommended three pentanomials for ECC implementation, such as the $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. A few efforts have been done on efficient realization of multiplication over $GF(2^m)$ based on irreducible pentanomials [9]-[18].

The systolic design usually has an efficient area-time implementation, being supported by its feature such as modularity and regularity of the structure; each processing element (PE) in the structure has the same or similar circuit design; one PE can pass the signals to its neighboring PE at a high speed since all PEs of the structure are pipelined [19].

Among all these designs, however, systolic realization (bit-parallel or digit-serial) of field multiplications over $GF(2^m)$ based on pentanomials are not so abundant. A systolic multiplier based on general polynomial is presented in [56]. Then, a semi-systolic multiplier is proposed in [57]. Two systolic multipliers have been proposed for error detection in [58] and [59], respectively. In a recent report [25], Meher has presented an efficient bit-parallel systolic structure for multiplication over $GF(2^m)$ based on irreducible pentanomial. A low-latency bit-parallel systolic multiplier is introduced in [60]. A novel low-latency Montgomery multiplier is newly reported in [61]. To achieve area-time tradeoff, digit-serial multipliers based on pentanomials are reported in [62] and [63], respectively. Very recently, low-latency digit-serial systolic multipliers are proposed in [64]. But the design strategy in [64] is only suitable for almost equally spaced polynomial (AESP) and can not be applied to NIST pentanomial-based multiplier. Generally, all the existing systolic multipliers, including bit-parallel and digit-serial structures, suffer several issues:

For bit-parallel systolic multipliers

- 1. Bit-parallel systolic structures usually have long latency and there are few reports about low-latency systolic realization
- 2. Critical-paths are large due to complexity of pentanomial based multiplication
- 3. Bit-parallel systolic structures usually have large register-complexity

For digit-serial systolic multipliers

- 1. Critical-paths of digit-serial systolic structures usually are a function of digit-size or field-order, which reduces throughput rate

- 2. Average computation time (ACT) of the structure increases with digit-size or field-order

Keeping these in view, in this chapter, we introduce three pairs of low-latency high-throughput bit-parallel and digit-serial systolic structures specifically for NIST pentanomials. A novel decomposition scheme is proposed first that we can decompose the multiplication into several parallel processing arrays to obtain a low-latency bit-parallel systolic structure with a critical-path of $2T_X$. These parallel arrays are then projected along vertical direction to obtain a digit-serial structure with the same critical-path. To increase the throughput rate, a novel modular reduction is introduced that the critical-paths of the bit-parallel and digit-serial structures are reduced to $(T_A + T_X)$. We have presented two modified bit-parallel structures with low area-complexity based on identical data sharing technique. Finally, we have proposed KA-based [65] bit-parallel and digit-serial multipliers to reduce the time complexity further.

The rest of the chapter is organized as follows. The proposed bit-parallel and digit-serial multipliers-I for finite field multiplication over $GF(2^m)$ based on irreducible pentanomials are presented in Section 4.2. In Section 4.3, the proposed bit-parallel and digit-serial multipliers-II are depicted. In Section 4.4, the proposed KA-based bit-parallel and digit-serial multipliers-III are proposed. In Section 4.5, the comparison and discussion of the hardware and time complexities are described. And the conclusion is given in Section 4.6.

4.2 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL MULTIPLIERS-I

4.2.1 Proposed algorithm

Let A, B and C are field elements in $GF(2^m)$, such that we can have the multiplication algorithm as

$$C = A \cdot B \bmod f(x) \tag{4.1}$$

where $A = \sum_{i=0}^{m-1} a_i x^i,\ B = \sum_{i=0}^{m-1} b_i x^i,\ C = \sum_{i=0}^{m-1} c_i x^i$, for $a_i,\ b_i$ and $c_i \in \{0,\ 1\}$. Then, we can express (4.1) in expanded form of

$$C = \sum_{i=0}^{m-1} a_i (B \cdot x^i \bmod f(x)) = \sum_{i=0}^{m-1} X_i = \sum_{i=0}^{m-1} B^i \cdot a_i \tag{4.2}$$

where $B^0 = B$, and $B^i = (B \cdot x^i \bmod f(x)) = \sum_{j=0}^{m-1} b_j^i x^i$.

Let w and d be two integers such that $m = wd + r$, where $0 \leq r < d$. For simplicity of discussion, we assume[1] $r = 0$, and decompose the input operand A into w number of bit-vectors A_u for $u = 0, 1, \cdots, w-1$, as follows:

$$A_u = [a_u\ a_{w+u} \cdots\ a_{m-w+u}] \tag{4.3}$$

Similarly, we can generate w number of operand vectors B_u for $u = 0, 1, \cdots, w-1$, as follows:

$$B_u = [B^u\ B^{w+u} \cdots\ B^{m-w+u}] \tag{4.4}$$

The product given in (4.1) can be decomposed into w inner- products of vectors A_u and B_u for $u = 0, 1, \cdots, w-1$ as:

$$\begin{aligned} C &= B_0 A_0^T + B_1 A_1^T + \cdots + B_{w-1} A_{w-1}^T \\ &= \sum_{u=0}^{w-1} B_u A_u^T = \sum_{u=0}^{w-1} \overline{C_u} \end{aligned} \tag{4.5}$$

where $\overline{C_u} = B_u A_u^T$. Note that each A_u for $u = 0, 1, \cdots, w-1$ is a d-point bit-vector and each B_u for $u = 0, 1, \cdots, Q-1$ is a d-term operand-vector. From (4.5) we can find that the desired multiplication can be performed by w cycles of successive accumulation of $\overline{C_u}$ for $u = 0, 1, \cdots, w-1$, while each $\overline{C_u}$ can be computed as $\overline{C_u} = \sum_{v=0}^{d-1} B^{u+vw} a_{u+vw}$.

Define field $GF(2^m)$ is constructed from pentanomial of degree m as $f(x) = x^m + x^{k1} + x^{k2} + x^{k3} + 1$, for $1 \leq k3 < k2 < k1 \leq m-1$. Then we can have B^1 from B as:

$$B^1 = B \cdot x \bmod f(x) = b_{m-1}^1 x^{m-1} + \ldots + b_1^1 x + b_0^1 \tag{4.6}$$

[1] When $r \neq 0$, we can append $(w - r)$ number of zeros to the operands to have $m = wd$.

$$
\begin{aligned}
b_0^1 &= b_{m-1} \\
b_{k3}^1 &= b_{k3-1} + b_{m-1} \\
b_{k2}^1 &= b_{k2-1} + b_{m-1} \\
b_{k1}^1 &= b_{k1-1} + b_{m-1} \\
b_j^1 &= b_{j-1}, \text{ for } j = 1, \ldots, m-1 \text{ and } j \neq k1, k2, k3
\end{aligned} \tag{4.7}
$$

And we can also obtain B^i from B for $i > 2$ for NIST pentanomials as (usually i is not a big number since we want to decompose the multiplier into i arrays):

$$
B^i = B \cdot x^i \bmod f(x) = b_{m-1}^i x^{m-1} + \ldots + b_1^i x + b_0^i \tag{4.8}
$$

where

$$
\begin{aligned}
b_0^i &= b_{m-i} \\
b_{k3+j}^i &= b_{m+k3-i+j} + b_{m-i+j}, \text{ for } 0 \leq j \leq k2-1-k3 \\
b_{k2+j}^i &= b_{m-i+k2+j} + b_{m-i-k3+k2+j} + b_{m-i+j}, \\
&\text{for } 0 \leq j \leq k1-1-k2 \\
b_{k1+j}^i &= b_{m-i+k1+j} + b_{m-i-k3+k1+j} + b_{m-i+k1-k2+j} \\
&+ b_{m-i+j}, \text{ for } 0 \leq j \leq i-1-k1 \\
b_{i+j}^i &= b_{0+j} + b_{m-k3+j} + b_{m-k2+j} + b_{m-k1+j}, \\
&\text{for } 0 \leq j \leq k3-1 \\
b_{k3+i+j}^i &= b_{k3+j} + b_{m-k2+k3+j} + b_{m-k1+k3+j}, \\
&\text{for } 0 \leq j \leq k2-k3-1 \\
b_{k2+i+j}^i &= b_{k1-1+j} + b_{m-k1+k2+j}, \text{ for } 0 \leq j \leq k1-k2-1 \\
b_j^i &= b_{j-i}, \text{ for } j = \text{others}
\end{aligned} \tag{4.9}
$$

Following (4.8) and (4.9), we can obtain B^w from B^0 (i is substituted as w). Moreover, we can extend (4.8) and (4.9) further to obtain $B^{(v+1)w+u}$ from B^{vw+u} for $v = 0, 1, \ldots, d-1$ and $u = 0, 1, \ldots, w-1$ (it is the similar process as (4.9)).

The proposed combined (bit-parallel and digit-serial) multiplication algorithm thus based on (4.5), (4.7)-(4.9) is described in Algorithm 4.1.

Algorithm 4.1 Proposed combined (bit-parallel and digit-serial) multiplication algorithm

---

Inputs: A and B are the pair of elements in $GF(2^m)$ to be multiplied.

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

1. Initialization step

   1.1 $D = 0$ (for digit-serial multiplication)

2. Multiplication step

   2.1. for $u = 0$ to $w - 1$

   2.2. for $v = 0$ to $d - 1$

   2.3-I. $C = \sum_{u=0}^{w-1} B_u A_u^T$ (for bit-parallel multiplication)

   2.3-II. $D = D + B_u A_u^T$ (for digit-serial multiplication)

   2.4. end for

   2.5. end for

3. Final step

   3.1. $C = D$ (for digit-serial multiplication)

---

where step 2.3-I and 2.3.-II refer to the digit-serial and bit-parallel multiplication processes, respectively. According to our proposed algorithm, for bit-parallel realization, we can have several arrays of partial products processed in parallel to lower the latency; while for digit-serial realization, we can have partial products accumulated as soon as they are computed, which shortens the ACT significantly.

To reduce the complexity of modular reduction operation, we introduce here an identical data sharing technique. Define x^m, $x^{m+1}, \ldots,$ x^{m+3w-2} as extended polynomial basis, and based on this definition we can write the equations into following steps (to reduce the register complexity in the structure, thus the whole area complexity of the structure will be

significantly reduced, and this proposed register-sharing technique can also be used in other structures):

$$B_{(E)}^{w-1} = \sum_{i=0}^{m+3w-2} b_i^{w-1} x^i = B^{w-1} + \sum_{i=m}^{m+3w-2} b_i^{w-1} x^i \tag{4.10}$$

where

$$\begin{aligned}
&\sum_{i=m}^{m+2k1-k2-2} b_i^{w-1} = \sum_{i=0}^{2k1-k2-2} b_i \\
&\sum_{i=m+2k1-k2-1}^{m+w+k1-k3-1} b_i^{w-1} = \sum_{i=m-w+k1-k2+k3-1}^{m-1} b_i \\
&\sum_{i=m+w+k1-k3}^{m+w+k1-k3+1} b_i^{w-1} = \sum_{i=1}^{2} b_{k2}^i \\
&\sum_{i=m+w+k1-k3+2}^{m+2w+k1-k3} b_i^{w-1} = \sum_{i=3}^{w-1} b_{k2-1}^i \\
&\sum_{i=m+2w+k1-k3+1}^{m+3w-2} b_i^{w-1} = \sum_{i=3}^{w-1} b_{k1-1}^i
\end{aligned} \tag{4.11}$$

where we can find that bits $\{b_i^{w-1}\ (0 \le i \le k3-1,\ k1+w-1 \le i \le m+w+k1-k3-1)\}$ can be selected to construct operand B^0, ... and bits $\{b_i^{w-1}(0 \le i \le m-1)\}$ can be selected to construct B^{w-1}, i.e., B^0 to B^{w-1} can be obtained through bit-select operations from $B_{(E)}^{w-1}$. And we can further define

$$\begin{aligned}
&B_{(E)bs\{0\}}^{w-1} = B^0 \\
&B_{\{E\}bs\{1\}}^{w-1} = B^1 \\
&\cdots\ \cdots\ \cdots \\
&B_{\{E\}bs\{w-1\}}^{w-1} = B^{w-1}
\end{aligned} \tag{4.12}$$

where $bs\{\cdot\}$ denotes the bit-select operations to obtain corresponding operand. This strategy can significantly reduce the register complexity in the systolic multiplier since many bits can be shared.

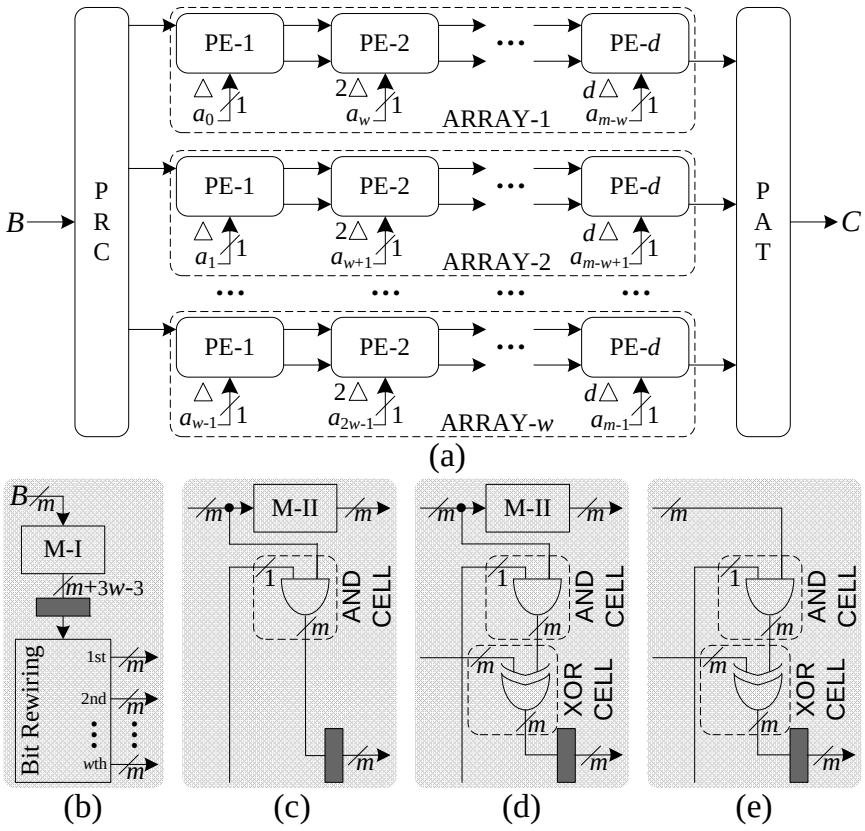

Figure 7: Proposed bit-parallel systolic multiplier-I (BP-I) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of PE-1. (d) Internal structure of regular PE. (e) Internal structure of PE-d.

4.2.2 Proposed bit-parallel systolic multiplier-I (BP-I)

The proposed bit-parallel systolic multiplier-I (BP-I) based on Algorithm 4.1 is shown in Fig. 7. It consists of one pre-computing (PRC) cell, one pipelined adder tree (PAT) cell and w systolic arrays (each array has d PEs). The PRC cell, as shown in Fig. 7(b), consists of a M-I cell and a bit rewiring cell, yields w outputs (B^0, B^1, ..., B^{w-1}) to corresponding arrays (M-I cell in PRC derives $B^{w-1}_{(E)}$ from B). The internal structure of PEs, i.e., PE-1, regular PE (PE-2 to PE-$(d-1)$ and PE-d, are shown in Fig. 7(c)-(e), respectively. A regular PE

consists of a M-II cell (M-II cell in PE-1 to PE-$(d-1)$ derives $B^{(v+1)w+u}$ from B^{vw+u} for $v = 0, 1, \ldots, d-1$ and $u = 0, 1, \ldots, w-1$), an AND cell and an XOR cell. During each cycle period, the result of AND cell is added together in XOR cell with another input from left and the result is then latched out to the right. Meanwhile, the output of M-II cell is latched out to the next PE to be used for the next cycle. Thus, critical-path of BP-I shall be $\max\{T_{PRC},\ T_{M-II},\ T_A + T_X\}$=$2T_X$ (M-II cell has a duration of $2T_X$ according to (9)), where T_{PRC} and T_{M-II} refer to the propagation time of PRC cell and M-II cell, respectively. BP-I yields its first output $(d+1+\log_2 w)$ cycles after the operands are fed to the structure, and the successive output will be available in every cycle.

4.2.3 Proposed modified BP-I (MBP-I)

To reduce the area-complexity of BP-I further, we can extend the strategy introduced in (4.10) and (4.11), identical data sharing technique, to all PEs of BP-I. Here we define as:

$$\begin{aligned} B_{(E)}^{(v+1)w+u} &= \sum_{i=0}^{m+3w-2} b_i^{(v+1)w+u} x^i \\ &= B^{(v+1)w+u} + \sum_{i=m}^{m+3w-2} b_i^{(v+1)w+u} x^i \end{aligned} \tag{4.13}$$

where

$$\begin{aligned} \sum_{i=m}^{m+2k1-k2-2} b_i^{(v+1)w+u} &= \sum_{i=0}^{2k1-k2-2} b_i^{(v+1)w} \\ \sum_{i=m+2k1-k2-1}^{m+w+k1-k3-1} b_i^{(v+1)w+u} &= \sum_{i=m-w+k1-k2+k3-1}^{m-1} b_i^{(v+1)w} \\ \sum_{i=m+w+k1-k3}^{m+w+k1-k3+1} b_i^{(v+1)w+u} &= \sum_{i=1}^{2} b_{k2}^{(v+1)w+i} \\ \sum_{i=m+w+k1-k3+2}^{m+2w+k1-k3} b_i^{(v+1)w+u} &= \sum_{i=3}^{w-1} b_{k2-1}^{(v+1)w+i} \\ \sum_{i=m+2w+k1-k3+1}^{m+3w-2} b_i^{(v+1)w+u} &= \sum_{i=3}^{w-1} b_{k1-1}^{(v+1)w+i} \end{aligned} \tag{4.14}$$

Similarly we can obtain $B^{(v+1)w+u}$ from $B_{(E)}^{(v+1)w+u}$ through bit-select operation, for $u = 0, 1, \ldots, w-1$. And we can also easily obtain $B_{(E)}^{(v+1)w+u}$ from $B_{(E)}^{vw+u}$:

For $b_i^{(v+1)w+u} x^i$ $(0 \le i \le m-1)$:

$$
\begin{aligned}
& b_0^{(v+1)w+u} = b_{m-w}^{vw+u} \\
& b_{k3+j}^{(v+1)w+u} = b_{m+k3-w+j}^{vw+u} + b_{m-w+j}^{vw+u} \text{for } 0 \le j \le k2-1-k3 \\
& b_{k2+j}^{(v+1)w+u} = b_{m-w+k2+j}^{vw+u} + b_{m-w-k3+k2+j}^{vw+u} + b_{m-w+j}^{vw+u} \\
& \text{for } 0 \le j \le k1-1-k2 \\
& b_{k1+j}^{(v+1)w+u} = b_{m-w+k1+j}^{vw+u} + b_{m-w-k3+k1+j}^{vw+u} \\
& + b_{m-w+k1-k2+j}^{vw+u} + b_{m-w+j}, \text{ for } 0 \le j \le i-1-k1 \\
& b_{w+j}^{(v+1)w+u} = b_{0+j}^{vw+u} + b_{m-k3+j}^{vw+u} + b_{m-k2+j}^{vw+u} + b_{m-k1+j}^{vw+u} \\
& \text{for } 0 \le j \le k3-1 \\
& b_{k3+w+j}^{(v+1)w+u} = b_{k3+j}^{vw+u} + b_{m-k2+k3+j}^{vw+u} + b_{m-k1+k3+j}^{vw+u} \\
& \text{for } 0 \le j \le k2-k3-1 \\
& b_{k2+w+j}^{(v+1)w+u} = b_{k1-1+j}^{vw+u} + b_{m-k1+k2+j}^{vw+u} \\
& 0 \le j \le k1-k2-1 \\
& b_j^{(v+1)w+u} = b_{j-w}^{vw+u}, \text{ for } j = \text{others}
\end{aligned}
\tag{4.15}
$$

For $b_i^{(v+1)w+u} x^i$ $(m \le i \le m+3w-2)$:

$$
\begin{aligned}
& \sum_{i=m}^{m+2k1-k2-2} b_i^{(v+1)w+u} = \sum_{i=0}^{2k1-k2-2} b_i^{vw+u} \\
& \sum_{i=m+2k1-k2-1}^{m+w+k1-k3-1} b_i^{(v+1)w+u} = \sum_{i=m-w+k1-k2+k3-1}^{m-1} b_i^{vw+u} \\
& \sum_{i=m+w+k1-k3}^{m+w+k1-k3+1} b_i^{(v+1)w+u} = \sum_{i=1}^{2} b_{k2}^{vw+u+i} \\
& \sum_{i=m+w+k1-k3+2}^{m+2w+k1-k3} b_i^{(v+1)w+u} = \sum_{i=3}^{w-1} b_{k2-1}^{vw+u+i} \\
& \sum_{i=m+2w+k1-k3+1}^{m+3w-2} b_i^{(v+1)w+u} = \sum_{i=3}^{w-1} b_{k1-1}^{vw+u+i}
\end{aligned}
\tag{4.16}
$$

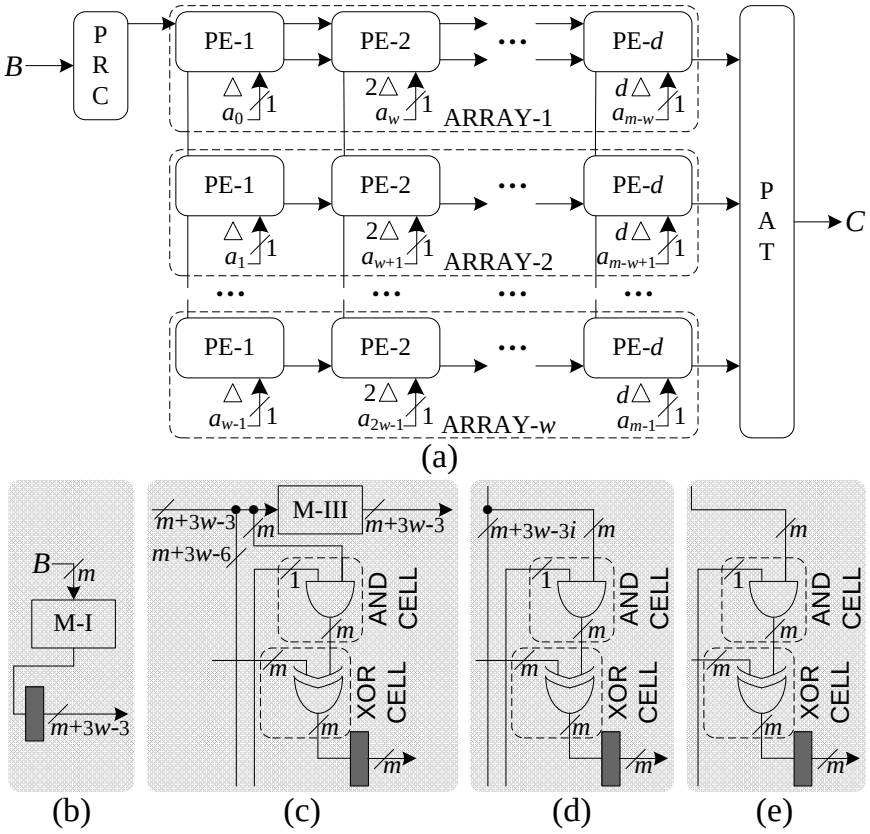

Figure 8: Proposed modified bit-parallel systolic multiplier-I (MBP-I) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of regular PE of Array-1. (d) Internal structure of regular PE of Array-2 to Array-$w-1$, where $2 \leq i \leq w-1$. (e) Internal structure of regular PE of Array-w.

where $\sum_{i=1}^{2} b_{k2}^{vw+u+i}$, $\sum_{i=3}^{w-1} b_{k2-1}^{vw+u+i}$ and $\sum_{i=3}^{w-1} b_{k1-1}^{vw+u+i}$ can be obtained from bits of operand B^{vw+u} according to (9).

Based on the above strategy, we can have modified BP-I (MBP-I) as shown in Fig. 8. The internal structures of PRC cell and regular PE of various arrays are shown in Fig. 8(b)-(e), respectively. M-III cell in regular PE of Array-1 derives $B_{(E)}^{(v+1)w+u}$ from $B_{(E)}^{vw+u}$, and $(m+3w-6)$ bits are spontaneously selected/shared by $(w-1)$ PEs in the same position (vertically in one column) of Array-2 to Array-w. The proposed data sharing strategy

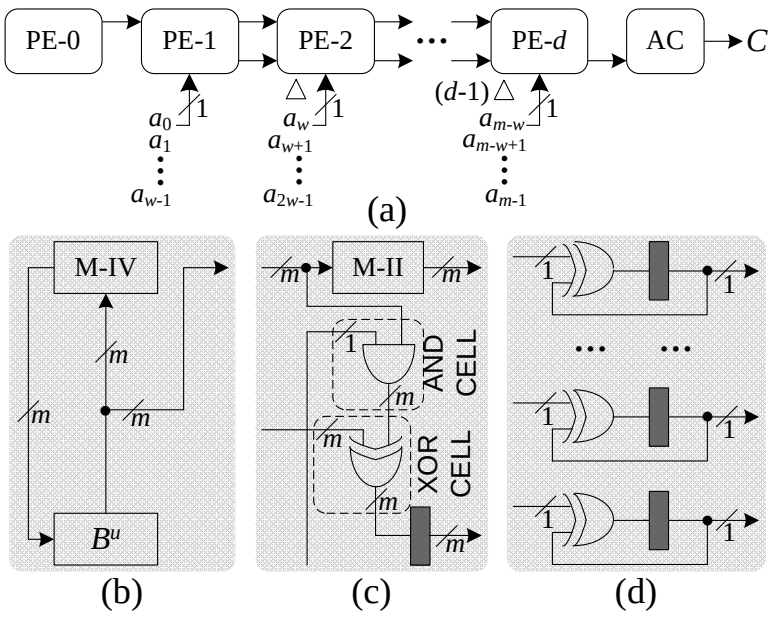

Figure 9: Proposed digit-serial systolic multiplier-I (DS-I), where Δ denotes unit delay and black block denotes register cell. (a) Proposed structure. (b) Internal structure of PE[0], where $0 \leq u \leq w-1$. (c) Internal structure of a regular PE. (e) Internal structure of AC cell.

significantly reduces XOR gate and register numbers Since there is no M-III cell in PEs except Array-1. Thus, the area-complexity of MBP-I is smaller than that of BP-I. While the critical-path and latency of MBP-I are exactly the same as BP-I.

4.2.4 Proposed digit-serial systolic multiplier-I (DS-I)

Based on proposed Algorithm 4.1, we can project these parallel arrays of BP-I along vertical direction to have the proposed digit-serial systolic multiplier-I (DS-I) as shown in Fig. 9. DS-I consists of $(m+1)$ PEs and one accumulation (AC) cell. The internal structures of PE-0, regular PE and AC cell are shown in Fig. 9(b)-(d), respectively. As shown in Fig. 9(b), all bits of operand B are pre-loaded in m bit-registers and then are latched out (meanwhile the m output bits are also yielded to the next PE) to the M-IV cell to perform modular operation

by one degree during each cycle period (obtain B^{u+1} from B^u, for $u = 0, 1, \ldots, w-1$). The m output bits of M-IV cell is then latched into registers to be used in next cycle period. The regular PE, from PE-2 to PE-$(d-1)$, contains a M-II cell, an AND cell, a XOR cell and a register cell, the same as that in BP-I. The AC cell, as shown in Fig. 9(d), contains m parallel bit-level finite field accumulators. During each cycle period, the newly received input is then added with the previously accumulated result and the result is stored in the register cell to be used during the next cycle. DS-I has the same critical-path as that of BP-I/MBP-I, and it gives the first output of desired product $(d+w)$ cycles after the pair of operands are fed to the structure, while the successive output are produced at the interval of w cycles thereafter.

4.3 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL SYSTOLIC MULTIPLIERS-II

4.3.1 Proposed algorithm

Both BP-I/MBP-I and DS-I have critical path of $2T_X$, which is also the duration time of M-I/M-II/M-III cell. To have a higher throughput rate design (lower critical-path), we need to reduce the duration time of M-I/M-II/M-III cell. In this section, we introduce a novel modular reduction operation that the delay time is reduced to T_X, and it can be extended further to apply the data sharing technique to reduce the area complexity of the structure.

Let us reconsider the operation of (4.9) (derive B^i from B for $i > 2$) first: we can define $g = (2i - k1 + k3)$ number of extended polynomial basis as x^m, x^{m+1}, …, x^{m+g-1}. Then we can define as

$$B^0_{\{P\}} = \sum_{j=0}^{m+g-1} b^0_{\{P\}j} x^j \tag{4.17}$$

$$
\begin{aligned}
&b^0_{\{P\}j} = b_j, \text{ for } 0 \le j \le m-1 \\
&b^0_{\{P\}j} = b_{m-i+k2+e} + b_{m-i-k3+k2+e} \ (0 \le e \le k1-1-k2) \\
&\text{for } m \le j \le m+k1-1-k2 \\
&b^0_{\{P\}j} = b_{m-i+k1+e} + b_{m-i-k3+k1+e} \ (0 \le e \le i-1-k1) \\
&\text{for } m+k1-k2 \le j \le m-k2+i-1 \\
&b^0_{\{P\}j} = b_{m-i+k1-k2+e} + b_{m-i+e} \ (0 \le e \le i-1-k1) \\
&\text{for } m-k2+i \le j \le m-k2-k1+2i-1 \\
&b^0_{\{P\}j} = b_e + b_{m-k3+e} \ (0 \le e \le k3-1) \\
&\text{for } m-k2-k1+2i \le j \le m-k2+g-1 \\
&b^0_{\{P\}j} = b_{m-k2+e} + b_{m-k1+e} \ (0 \le e \le k3-1) \\
&\text{for } m-k2+g \le j \le m-k2+g+k3-1 \\
&b^0_{\{P\}j} = b_{m-k2+k3+e} + b_{m-k1+k3+e} \ (0 \le e \le k2-k3-1) \\
&\text{for } m-k2+g+k3 \le j \le m+g-1
\end{aligned}
\tag{4.18}
$$

Then, if we obtain B^i from $B^0_{\{P\}}$, we can have

$$
\begin{aligned}
&b^i_0 = b_{\{P\}m-i} \\
&b^i_{k3+j} = b_{\{P\}m+k3-i+j} + b_{\{P\}m-i+j}, \ 0 \le j \le k2-1-k3 \\
&b^i_{k2+j} = b_{\{P\}m+j} + b_{m-i+j}, \ 0 \le j \le k1-1-k2 \\
&b^i_{k1+j} = b_{\{P\}m+k1-k2+j} + b_{\{P\}m-k2+i+j}, \ 0 \le j \le i-1-k1 \\
&b^i_{i+j} = b_{\{P\}m-k2-k2+2i+j} + b_{\{P\}m-k2+g+j}, \ 0 \le j \le k3-1 \\
&b^i_{k3+i+j} = b_{k3+j} + b_{\{P\}m-k2+g+k3+j}, \ 0 \le j \le k2-k3-1 \\
&b^i_{k2+i+j} = b_{k1-1+j} + b_{m-k1+k2+j}, \ 0 \le j \le k1-k2-1 \\
&b^i_j = b_{\{P\}j-i}, \ j = \text{others}
\end{aligned}
\tag{4.19}
$$

where we define the operation of (4.19) as novel modular reduction operation, and the modular reduction time is reduced from $2T_X$ to T_X.

Similarly, we can define $B^i_{\{P\}} = \sum_{j=i}^{m+g-1} b^i_{\{P\}j} x^j$ as (4.17), and thus we can get B^{2i} from $B^i_{\{P\}}$ as similar operation as (4.19) in a duration of T_X.

We can also extend further the operations of (4.18) and (4.19), i.e., combine the operations of (4.18) and (4.19) together to derive $B^i_{\{P\}}$ from $B^0_{\{P\}}$ as follows:

For $B^i_{\{P\}}$ $(0 \le j \le m-1)$, we have similar operation as (4.19):

$$
\begin{aligned}
& b^i_{\{P\}0} = b_{\{P\}m-i} \\
& b^i_{\{P\}k3+j} = b_{\{P\}m+k3-i+j} + b_{\{P\}m-i+j},\ 0 \le j \le k2-1-k3 \\
& b^i_{\{P\}k2+j} = b_{\{P\}m+j} + b_{m-i+j},\ 0 \le j \le k1-1-k2 \\
& b^i_{\{P\}k1+j} = b_{\{P\}m+k1-k2+j} + b_{\{P\}m-k2+i+j},\ 0 \le j \le i-1-k1 \\
& b^i_{\{P\}i+j} = b_{\{P\}m-k2-k2+2i+j} + b_{\{P\}m-k2+g+j},\ 0 \le j \le k3-1 \\
& b^i_{\{P\}k3+i+j} = b_{k3+j} + b_{\{P\}m-k2+g+k3+j},\ 0 \le j \le k2-k3-1 \\
& b^i_{\{P\}k2+i+j} = b_{k1-1+j} + b_{m-k1+k2+j},\ 0 \le j \le k1-k2-1 \\
& b^i_{\{P\}j} = b_{\{P\}j-i},\ j = \text{others}
\end{aligned}
\tag{4.20}
$$

For $B^i_{\{P\}}$ $(m \le j \le m+g-1)$, we have similar operation as (4.18):

$$
\begin{aligned}
& b^i_{\{P\}j} = b_{\{P\}m-i+k2+e} + b_{\{P\}m-i-k3+k2+e}\ (0 \le e \le k1-1-k2) \\
& \text{for } m \le j \le m+k1-1-k2 \\
& b^i_{\{P\}j} = b_{\{P\}m-i+k1+e} + b_{\{P\}m-i-k3+k1+e}\ (0 \le e \le i-1-k1) \\
& \text{for } m+k1-k2 \le j \le m-k2+i-1 \\
& b^i_{\{P\}j} = b_{\{P\}m-i+k1-k2+e} + b_{\{P\}m-i+e}\ (0 \le e \le i-1-k1) \\
& \text{for } m-k2+i \le j \le m-k2-k1+2i-1 \\
& b^i_{\{P\}j} = b_{\{P\}e} + b_{\{P\}m-k3+e}\ (0 \le e \le k3-1) \\
& \text{for } m-k2-k1+2i \le j \le m-k2+g-1 \\
& b^i_{\{P\}j} = b_{\{P\}m-k2+e} + b_{\{P\}m-k1+e}\ (0 \le e \le k3-1) \\
& \text{for } m-k2+g \le j \le m-k2+g+k3-1 \\
& b^i_{\{P\}j} = b_{\{P\}m-k2+k3+e} + b_{\{P\}m-k1+k3+e}\ (0 \le e \le k2-k3-1) \\
& \text{for } m-k2+g+k3 \le j \le m+g-1
\end{aligned}
\tag{4.21}
$$

and we can extend further to derive $B_{\{P\}}^{(v+1)w+u}$ from $B_{\{P\}}^{vw+u}$ for $v = 0, 1, \ldots, d-1$ and $u = 0, 1, \ldots, w-1$. Define again as $B_{bs\{P\}}^{i} = B^i$, where bs refers to bit-select operation of $B_{\{P\}}^{i}$ to obtain B^i, then (4.5) can be rewritten as:

$$\begin{aligned} C &= B_0 A_0^T + B_1 A_1^T + \cdots + B_{w-1} A_{w-1}^T \\ &= \sum_{u=0}^{w-1} B_u A_u^T = \sum_{u=0}^{w-1} \sum_{v=0}^{d-1} B_{bs\{P\}}^{u+vw} a_{u+vw} = \sum_{u=0}^{w-1} \overline{C_u} \end{aligned} \tag{4.22}$$

where $\overline{C_u} = \sum_{v=0}^{d-1} B_{bs\{P\}}^{u+vw} a_{u+vw}$.

The proposed combined (bit-parallel and digit-serial) multiplication algorithm 2 thus based on (4.18)-(4.22) is described in Algorithm 4.2.

Algorithm 4.2 Proposed combined (bit-parallel and digit-serial) multiplication algorithm

---

Inputs: A and B are the pair of elements in $GF(2^m)$ to be multiplied.

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

1. Initialization step

   1.1 $D = 0$ (for digit-serial multiplication)

2. Multiplication step

   2.1. for $u = 0$ to $w-1$

   2.2. for $v = 0$ to $d-1$

   2.3-I. $C = \sum_{u=0}^{w-1} B_u A_u^T = \sum_{u=0}^{w-1} \sum_{v=0}^{d-1} B_{bs\{P\}}^{u+vw} a_{u+vw}$ (for bit-parallel multiplication)

   2.3-II. $D = D + B_u A_u^T = D + \sum_{v=0}^{d-1} B_{bs\{P\}}^{u+vw} a_{u+vw}$ (for digit-serial multiplication)

   2.4. end for

   2.5. end for

3. Final step

   3.1. $C = D$ (for digit-serial multiplication)

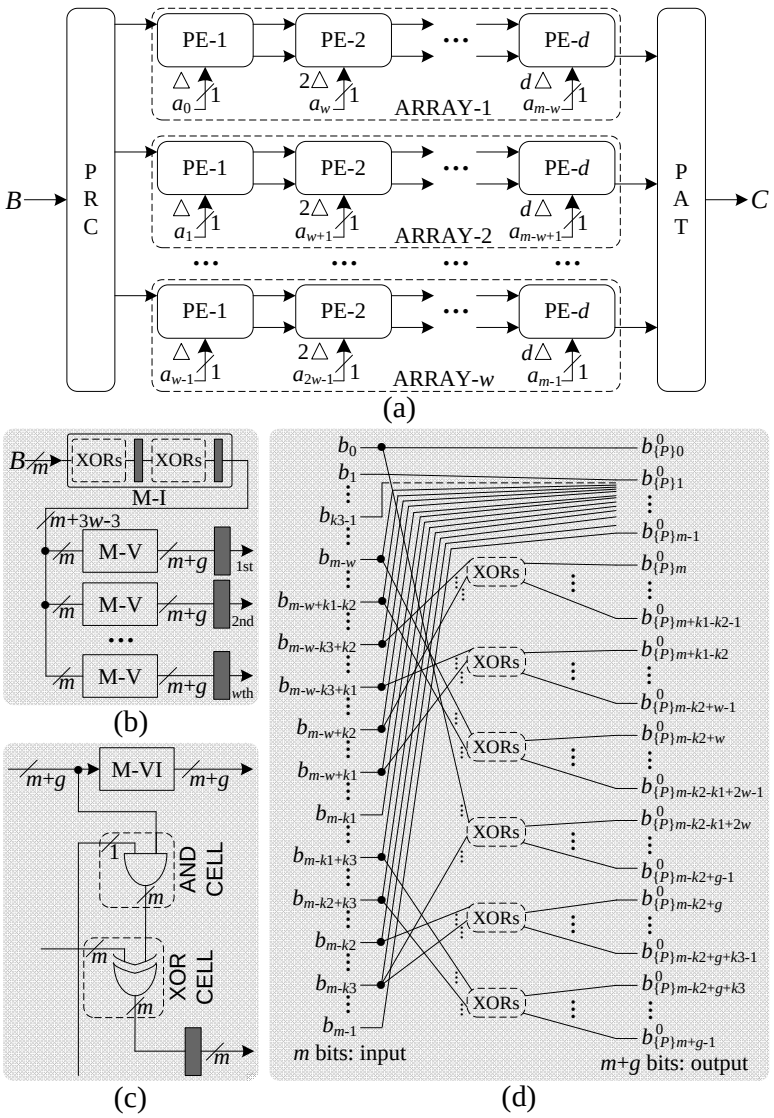

Figure 10: Proposed bit-parallel systolic multiplier-II (BP-II), where Δ denotes unit delay and black block denotes register cell $g = (2w - k1 + k3)$. (a) Proposed structure. (b) Internal structure of PRC, where M-I cell is designed into two-stage pipeline to reduce the critical-path to T_X. (c) Internal structure of a regular PE. (d) Detailed structure of M-V cell in PRC (derives $B^0_{\{P\}}$ from B^0).

where step 2.3-I refers to the digit-serial multiplication process and step 2.3.-II refers to the bit-parallel multiplication process. According to our proposed algorithm, for both bit-

parallel and digit-serial realizations, the duration time of novel modular reduction operation is reduced to T_X, which increases throughput rate of the proposed structures based Algorithm 4.2.

4.3.2 Proposed bit-parallel systolic multiplier-II (BP-II)

The proposed bit-parallel systolic multiplier-II (BP-II) based on Algorithm 4.2 is shown in Fig. 10. BP-II has nearly the same structure as BP-I except the internal structures of PRC and PE. The internal structure of PRC is shown in Fig. 10(b), where it yields w outputs to corresponding w arrays. To maintain the critical-path of PRC as T_X, we have also used a two-stage pipelined M-I cell to realize the operation of (4.10). The internal structure of regular PE is shown in Fig. 10(c). Fig. 10(d) gives the detail design of M-V cell in PRC to derive $B^0_{\{P\}}$ from B^0, while another novel modular reduction cells, M-VI, have similar structures to realize similar operations as (4.20) and (4.21). The time duration of M-V cell and M-VI cell, according to (4.18), (4.20) and (4.21), is T_X. BP-II thus has a critical-path of $\max\{T_{M-V}, T_{M-VI}, T_A+T_X\}=T_A+T_X$ (M-I cell is two-stage pipelined and thus critical-path of PRC is T_X), where T_{M-V} and T_{M-VI} refer to the propagation time of M-V cell and M-VI cell, respectively. BP-II yields its first output $(d+1+\log_2 w)$ cycles after the operands are fed to the structure, while the successive output can be obtained in every cycle thereafter. BP-II, therefore, has higher throughput rate than BP-I at the cost of some small number of XOR and registers. The overhead of hardware complexity, however, is minor when compare to the achievement in throughput rate increase, especially for those high-throughput application systems.

4.3.3 Proposed modified BP-II (MBP-II)

To reduce the area-complexity of BP-II further, we can follow the data sharing operations of (4.13), and we can define as follows (where $g = (2w - k1 + k3)$):

We first define $(3w - 3 + g + w - 1 = 4w + g - 4)$ number of extended polynomial basis as x^m, x^{m+1}, ..., $x^{m+4w+g-3}$. Then we can have the following steps based on the above definition (likewise, this proposed strategy can significantly reduce the area-complexity of the

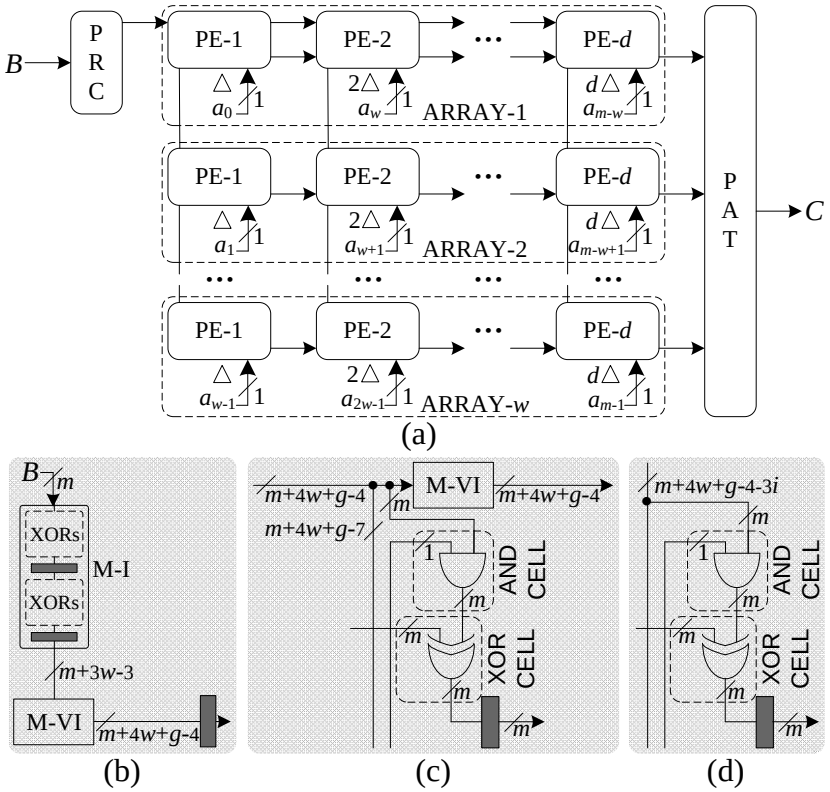

Figure 11: Proposed modified bit-parallel systolic multiplier-II (MBP-II) over $GF(2^m)$, where $\triangle$ denotes a unit delay and black block denotes register cell $g = (2w-k1+k3)$. (a) Proposed systolic structure. (b) Internal structure of PRC cell. (c) Internal structure of regular PE of Array-1. (d) Internal structure of regular PE of Array-2 to Array-$w-1$, where $2 \leq i \leq w-1$.

structure, especially the register complexity, and this technique can be extended to various applications):

$$\begin{aligned} B_{(NE)}^{(v+1)w+u} &= \sum_{i=0}^{m+3w-2+g+w-1} b_i^{(v+1)w+u} x^i = \sum_{i=0}^{m+3w-2} b_i^{(v+1)w+u} x^i \\ &+ \sum_{i=m+3w-3}^{m+3w-2+g} b_i^{(v+1)w+u} x^i + \sum_{i=m+3w-2+g+1}^{m+3w-2+g+w-1} b_i^{(v+1)w+u} x^i = B_{(E)}^{(v+1)w+u} \\ &+ \sum_{i=m+3w-3}^{m+3w-2+g} b_i^{(v+1)w+u} x^i + \sum_{i=m+3w-1+g}^{m+4w+g-3} b_i^{(v+1)w+u} x^i \end{aligned} \tag{4.23}$$

$$\sum_{i=m+3w-3}^{m+3w-2+g} b_i^{(v+1)w+u} = \sum_{i=m}^{m+g-1} b_{\{P\}i}^{(v+1)w+u}$$
$$\sum_{i=m+3w-1+g}^{m+4w+g-3} b_i^{(v+1)w+u} = \sum_{i=0}^{w-1} b_{\{P\}m-k2+w}^{vw+u+i} \tag{4.24}$$
$$= \sum_{i=0}^{w-1} b_{m-w+k1-k2}^{vw+u+i} + b_{m-w}^{vw+u+i}$$

for $0 \leq u \leq w-1$.

Thus we can obtain $B_{(P)}^{vw+u+i}$ from $B_{(NE)}^{(v+1)w+u}$ through bit-select operation, for $i = 0, 1, \ldots, w-1$. And we can also easily obtain $B_{(NE)}^{(v+1)w+u}$ from $B_{(NE)}^{vw+u}$:

- Derive $B_{(E)}^{(v+1)w+u}$: $b_i^{(v+1)w+u}$ $(0 \leq i \leq m+3w-2)$ from $B_{(E)}^{vw+u}$ according to (4.15) and (4.16)
- Derive $b_i^{(v+1)w+u}$ $(m+3w-3 \leq i \leq m+3w-2+g)$ from $B_{(P)}^{vw+u}$ according to (4.20) and (4.21)
- Derive $b_i^{(v+1)w+u}$ $(m+3w-1+g \leq i \leq m+4w+g-3)$ from $B^{vw+u+j}(0 \leq j \leq w-1)$ according to (4.18) and (4.24)

Based on the above derivation, we can have modified BP-II (MBP-II) as shown in Fig. 11. The internal structures of PRC cell and regular PE of various arrays are shown in Fig. 11(b)-(d), respectively. The critical-path and latency of MBP-II are exactly the same as BP-II. M-VI cell in regular PE of Array-1 derives $B_{(NE)}^{(v+1)w+u}$ from $B_{(NE)}^{vw+u}$. Since there is no M-VI cell in PEs of Array-2 to Array-w, the area-complexity of MBP-II is smaller than that of BP-II. Because of its efficiency in area-complexity and high-throughput ability, it can be used in various environments with requirement of low area-complexity and high-throughput.

4.3.4 Proposed digit-serial systolic multiplier-II (DS-II)

Based on proposed Algorithm 4.2, we can have the proposed digit-serial systolic multiplier-II (DS-II) as shown in Fig. 12. DS-II consists of $(m+1)$ PEs and one accumulation (AC) cell. The internal structures of PE-0, regular PE and AC cell are shown in Fig. 12(b)-(d), respectively. As shown in Fig. 12(b), all bits of operand B are fed to the M-V cell and the

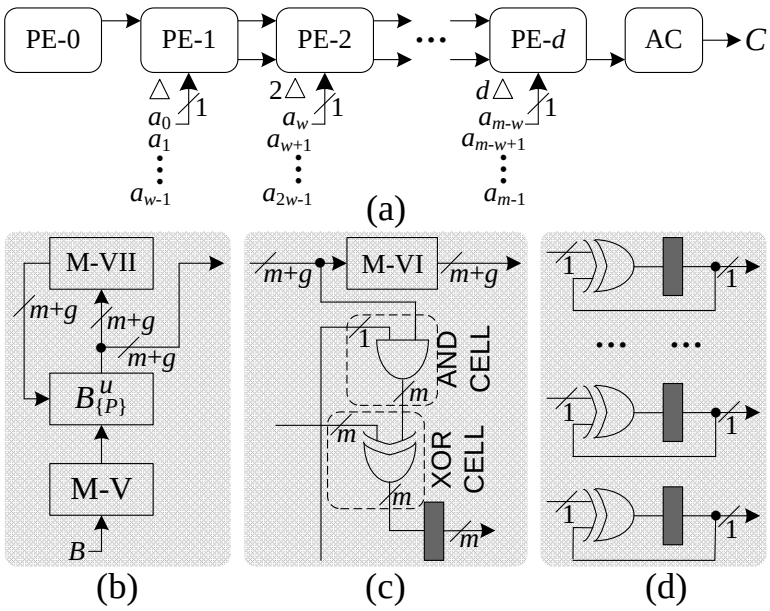

Figure 12: Proposed digit-serial systolic multiplier-II (DS-II), where Δ denotes unit delay and black block denotes register cell $g = (2w - k1 + k3)$. (a) Proposed structure. (b) Internal structure of PE[0], where $0 \leq u \leq w - 1$. (c) Internal structure of a regular PE. (d) Internal structure of AC cell.

output is then loaded in $(m+g)$ bit-registers and then are latched out (meanwhile the $(m+g)$ output bits are also yielded to the next PE) to the M-VII cell to perform modular operation by one degree during each cycle period (obtain $B^{u+1}_{\{P\}}$ from $B^{u}_{\{P\}}$, for $u = 0, 1, \ldots, w-1$). The $(m+g)$ output bits of M-VII cell are then latched into registers to be used in next cycle period. The regular PE, from PE-2 to PE-$(d-1)$, contains a M-VI cell, an AND cell, a XOR cell and a register cell, the same as that in BP-II. The AC cell, as shown in Fig. 12(d), contains m parallel bit-level finite field accumulators. During each cycle period, the newly received input is then added with the previously accumulated result and the result is stored in the register cell to be used during the next cycle. DS-II has the same critical-path as that of BP-II, and it gives the first output of desired product $(d+w)$ cycles after the pair of operands are fed to the structure, while the successive output are produced at the interval of w cycles thereafter.

4.4 PROPOSED BIT-PARALLEL AND DIGIT-SERIAL SYSTOLIC MULTIPLIERS-III

KA-based digit-level systolic multiplier is reported in a newly reported paper [18]. The structure presented in [18], however, is only suitable for AESP rather than NIST pentanomials. Therefore, in this section, we present a pair of bit-parallel and digit-serial systolic multipliers based on Karatsuba decomposition to achieve more efficiency in time-complexity.

4.4.1 Proposed algorithm

For simplicity of discussion, we present here a bit-parallel/digit-serial systolic multiplier based on two-term Karatsuba decomposition, which can be easily extended to n-term Karatsuba decomposition.

For two-term KA [20], element $A = \sum_{i=0}^{m-1} a_i x^i$ ($a_i \in GF(2)$) can be expressed as:

$$A = A_L^2 + A_H^2 x \tag{4.25}$$

where

$$\begin{aligned} A_L &= \sum_{j=0}^{\lceil m/2-1 \rceil} a_{2j} x^j \\ A_H &= \sum_{j=0}^{\lceil m/2-1 \rceil} a_{2j+1} x^j \end{aligned} \tag{4.26}$$

Similarly, $B = \sum_{i=0}^{m-1} b_i x^i$ ($b_i \in GF(2)$) can be expressed as $B = B_L^2 + B_H^2 x$. Product of A and B then is:

$$\begin{aligned} C = AB \bmod f(x) &= (A_L^2 + A_H^2 x)(B_L^2 + B_H^2 x) \\ &= (A_L B_L)^2 (1+x) + (A_L + A_H)^2 (B_L + B_H)^2 x \\ &+ (A_H B_H)^2 (x^2 + x) \bmod f(x) \end{aligned} \tag{4.27}$$

We define three partial products of (27) as: $C_L = A_L B_L$, $C_{LH} = (A_L + A_H)(B_L + B_H)$ and $C_H = A_H B_H$. For simplicity of discussion, we introduce here the proposed algorithm to

obtain $A_L B_L$, and it can be extended to obtain other two partial products: $(A_L + A_H)(B_L + B_H)$ and $A_H B_H$. Let C_L be expressed as

$$C_L = \sum_{i=0}^{\lceil m/2-1 \rceil} a_{2i}(B_L \cdot x^i) = \sum_{i=0}^{\lceil m/2-1 \rceil} X_{L,i} \tag{4.28}$$

where $X_{L,i} = a_{2i} \cdot B_L^i$, for $B_L^0 = B_L,$ and $B_L^i = B_L \cdot x^i$.

Define $m/2 = wd$, we can decompose operand A_L into w number of bit-vectors $A_{L,u}$ as $A_{L,u} = [a_{2u}\ a_{2w+2u}\ \cdots\ a_{m-2w+2u}]$ for $u = 0, 1, \cdots, w-1$. We can also generate w number of operand vectors $B_{L,u} = [B_L^u\ B_L^{w+u}\ \cdots\ B_L^{m-w+u}]$ for $u = 0, 1, \cdots, w-1$.

Equation (4.28) can then be expressed as:

$$\begin{aligned} C_L &= B_{L,0}A_0^T + B_{L,1}A_1^T + \cdots + B_{L,w-1}A_{w-1}^T \\ &= \sum_{u=0}^{w-1} B_{L,u}A_{L,u}^T = \sum_{u=0}^{w-1} \overline{C}_{L,u} \end{aligned} \tag{4.29}$$

where $\overline{C}_{L,u} = B_{L,u}A_{L,u}^T$, and each $\overline{C}_{L,u}$ can be obtained as

$$\overline{C}_{L,u} = \sum_{v=0}^{d-1} B_L^{wv+u} a_{2wv+2u} \tag{4.30}$$

For one specific value of v, we can derive B_L^{wv+u+s} from B_L^{wv+u}, for $1 \le s \le d$ as

$$B_L^{wv+u+s} = B_L^{wv+u} x^s \tag{4.31}$$

while B_L^{wv+u} can also be obtained directly from B_L^0 as $B_L^{wv+u} = B_L^0 x^{wv+u}$.

The proposed combined (bit-parallel and digit-serial) algorithm based on two-term Karatsuba decomposition (4.29)-(4.31) is described in Algorithm 4.3.

Algorithm 4.3 Proposed combined (bit-parallel and digit-serial) algorithm based on two-term Karatsuba decomposition

---

Inputs: A and B are the pair of elements in $GF(2^m)$.

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

1. Initialization step

1.1 $D_L = 0$, $D_{LH} = 0$, and $D_H = 0$ (for digit-serial multiplication)

1.2 Decompose as $A_{L,u} = [a_{2u} \cdots a_{m-2w+2u}]$, $B_{L,u} = [B_L^u \cdots B_L^{m-w+u}]$, similar operations to $A_L + A_H$, $B_L + B_H$, A_H and B_H

2. Multiplication step

2.1. for $u = 0$ to $w - 1$

2.2. for $v = 0$ to $d - 1$

2.3.1 $\overline{C}_{L,u} = \sum_{v=0}^{d-1} B_L^{wv+u} a_{2wv+2u}$

2.3.1-I $C_L = \sum_{u=0}^{w-1} \overline{C}_{L,u}$ (for bit-parallel multiplication) 2.3.1-II $D_L = D_L + \overline{C}_{L,u}$ (for digit-serial multiplication)

2.3.2-I $C_{LH} = \sum_{u=0}^{w-1} \overline{C}_{LH,u}$ (similar as 2.3.1-I, for bit-parallel multiplication)

2.3.2-II $D_{LH} = D_{LH} + \sum_{u=0}^{w-1} \overline{C}_{LH,u}$ (similar as 2.3.1-II, for digit-serial multiplication)

2.3.3-I $C_H = \sum_{u=0}^{w-1} \overline{C}_{H,u}$ (similar as 2.3.1-I, for bit-parallel multiplication)

2.3.3-II $D_H = D_H + \overline{C}_{H,u}$ (similar as 2.3.1-II, for digit-serial multiplication)

2.4. end for

2.5. end for

3. Final step

3.1. $C = C_L^2(1 + x) + C_{LH}^2 x + C_H^2(x^2 + x) \bmod f(x)$ (for bit-parallel multiplication)

3.1. $C = D_L^2(1 + x) + D_{LH}^2 x + D_H^2(x^2 + x) \bmod f(x)$ (for digit-serial multiplication)

---

where steps 2.3.1 to 2.3.3 refer to the parallel bit-parallel/digit-serial multiplication process.

The proposed multiplication algorithm based on two-term KA, however, can be extended to n-term Karatsuba decomposition.

Step 3.1 of Algorithm 4.3 can be obtained as follows:

$$\begin{aligned} C &= C_L^2(1 + x) + C_{LH}^2 x + C_H^2(x^2 + x) \bmod f(x) \\ &= C_L^2(1 + x) \bmod f(x) + C_{LH}^2 x \bmod f(x) \\ &\quad + C_H^2(x^2 + x) \bmod f(x) \end{aligned} \tag{4.32}$$

For simplicity of discussion, here we only cover the steps to obtain $C_L^2(1 + x) \bmod f(x)$, and it is similar steps to obtain $C_{LH}^2 x \bmod f(x)$ and $C_H^2(x^2 + x) \bmod f(x)$ (In step 3.1 of

Algorithm 3, $C_L^2(1+x)$ is equal to $D_L^2(1+x)$, as well as the other two partial terms). Let us define C_L as follows:

$$C_L = \sum_{i=0}^{m-1} c_{L,i} x^i \tag{4.33}$$

where $c_{L,i} \in GF(2)$. Such that we can have

$$\begin{aligned} C_L^2 &= c_{L,0} + c_{L,1}x^2 + c_{L,2}x^4 + \cdots + c_{L,m-1}x^{2m-2} \\ &= \sum_{i=0}^{m-1} c_{L,i} x^{2i} \end{aligned} \tag{4.34}$$

And we can further have

$$\begin{aligned} C_L^2(1+x) &= \sum_{i=0}^{m-1} c_{L,i} x^{2i}(1+x) \\ &= c_{L,0} + c_{L,0}x + c_{L,1}x^2 + c_{L,1}x^3 + c_{L,2}x^4 \\ &+ c_{L,2}x^5 + \cdots + c_{L,m-1}x^{2m-2} + c_{L,m-1}x^{2m-1} \\ &= \sum_{i=0}^{m-1} c_{L,\lfloor i/2 \rfloor} x^i + x^m \cdot \sum_{i=0}^{m-1} c_{L,(m-1)/2+\lfloor i/2 \rfloor} x^i \end{aligned} \tag{4.35}$$

Since x is the root of $f(x)$ ($f(x)$ is a pentanomial of degree m as $f(x) = x^m + x^{k1} + x^{k2} + x^{k3} + 1$, for $1 \leq k3 < k2 < k1 \leq m-1$). Thus we can have

$$f(x) = x^m + x^{k1} + x^{k2} + x^{k3} + 1 = 0 \tag{4.36}$$

Then, we can have

$$x^{k1} + x^{k2} + x^{k3} + 1 = x^m \tag{4.37}$$

Replace x^m in (4.27) with (4.29), and $C_L^2(1+x) \bmod f(x)$ can be computed as

$$\begin{aligned} C_L^2(1+x) \bmod f(x) &= \sum_{i=0}^{m-1} c_{L,\lfloor i/2 \rfloor} x^i \bmod f(x) \\ &+ x^m \cdot \sum_{i=0}^{m-1} c_{L,(m-1)/2+\lfloor i/2 \rfloor} x^i \bmod f(x) \\ &= C_{L1}^2 + (x^{k1} + x^{k2} + x^{k3} + 1) C_{L2}^2 \\ &= C_{L1}^2 + C_{L2}^2 + C_{L2}^2 \cdot x^{k1} + C_{L2}^2 \cdot x^{k2} + C_{L2}^2 \cdot x^{k3} \end{aligned} \tag{4.38}$$

where $C^2_{L1} = \sum_{i=0}^{m-1} c_{L,\lfloor i/2 \rfloor} x^i \bmod f(x)$ and $C^2_{L2} = \sum_{i=0}^{m-1} c_{L,(m-1)/2+\lfloor i/2 \rfloor} x^i \bmod f(x)$. And according to (4.8) and (4.9), we can obtain $C^2_{L2} \cdot x^{k1}$, $C^2_{L2} \cdot x^{k2}$ and $C^2_{L2} \cdot x^{k3}$ directly from C^2_{L2}, respectively, for NIST pentanomials. Similarly, we can have nearly the same steps to obtain $C^2_{LH} x \bmod f(x)$ and $C^2_H(x^2 + x) \bmod f(x)$.

4.4.2 Proposed bit-parallel systolic multiplier-III (BP-III)

The proposed bit-parallel systolic multiplier-III (BP-III) based on Algorithm 4.3 is thus shown in Fig. 13. It consists of three systolic-array modules (one gray box is one module), where each module consists of w systolic arrays (each array consists of d PEs). BP-III also has one pre-computing-adder (PCA) cell and one shift pipelined adder tree (SPT) cell. Besides, a final modular addition (FMA) cell is needed for the operation of step 3.1 of Algorithm 4.3. The internal structure of a regular PE (PE-2 to PE-d) is shown in Fig. 13(b). Comparing with regular PE, PE-1 does not have XOR cell. The internal structures of SPT and PCA are shown in Fig. 13(c) and (d), respectively. The design detail of FMA is shown in Fig. 13(e), where step 3.1 of Algorithm 4.2 is executed to obtain the output. There are two PAT cells used in FMA to maintain the systolic pipeline. It is noted that the internal structures of cell deriving $C^2_{LH} x \bmod f(x)$ and $C^2_H(x^2+x) \bmod f(x)$ are similar to structure of deriving $C^2_L(1 + x) \bmod f(x)$. The critical-path of proposed structure is $(T_A + T_X)$, which is the same as that of BP-II. The proposed design gives the first output $(d + \log_2 w + 6)$ cycles after operands are fed to the structure, while the successive outputs are obtained in every cycle thereafter.

4.4.3 Proposed digit-serial systolic multiplier-III (DS-III)

The proposed digit-serial systolic multiplier-III (DS-III) based on Algorithm 4.3 is shown in Fig. 14. It consists of three systolic arrays, where each array consists of d PEs and one shift-accumulation (SAC) cell. The internal structures of PCA cell, SPT cell and FMA cell are exactly the same as those of BP-III. The critical-path of proposed structure is $(T_A + T_X)$, which is the same as that of BP-III. The proposed design gives the first output $(d + w + 6)$ cycles after operands are fed to the structure, while the successive outputs are obtained

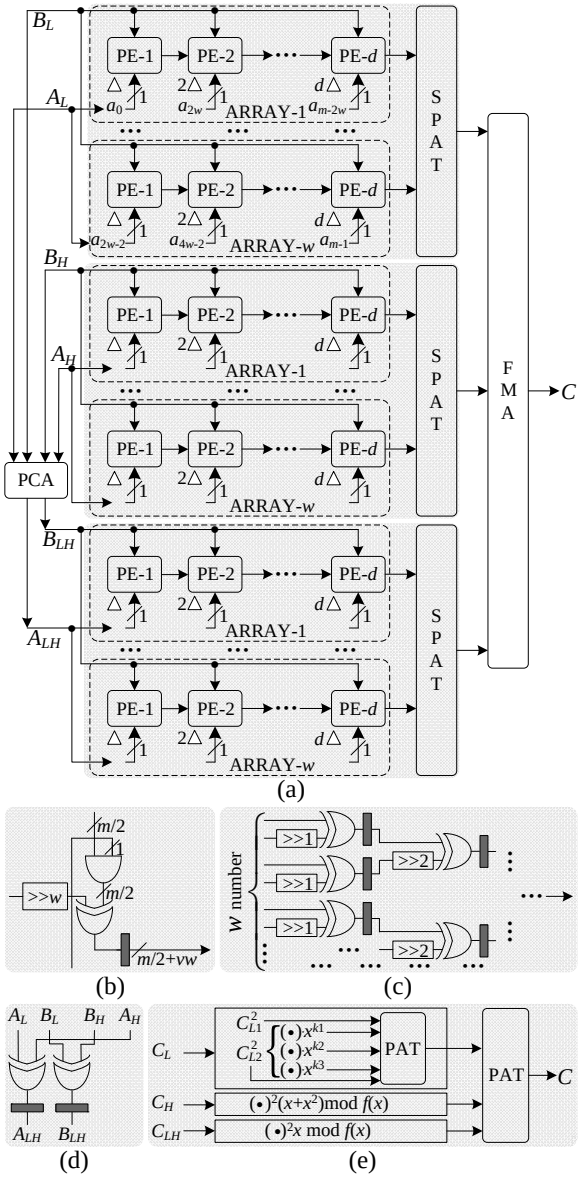

Figure 13: Proposed bit-parallel systolic multiplier-III (BP-III), where Δ denotes unit delay and black block denotes register cell. (a) Proposed structure, where each gray box represents a systolic-array module. (b) Internal structure of a regular PE, where $1 \leq v \leq d-1$ (there is no XOR cell in PE-1). (c) Internal structure of SPT. (d) Internal structure of PCA.(e) Internal structure of FMA.

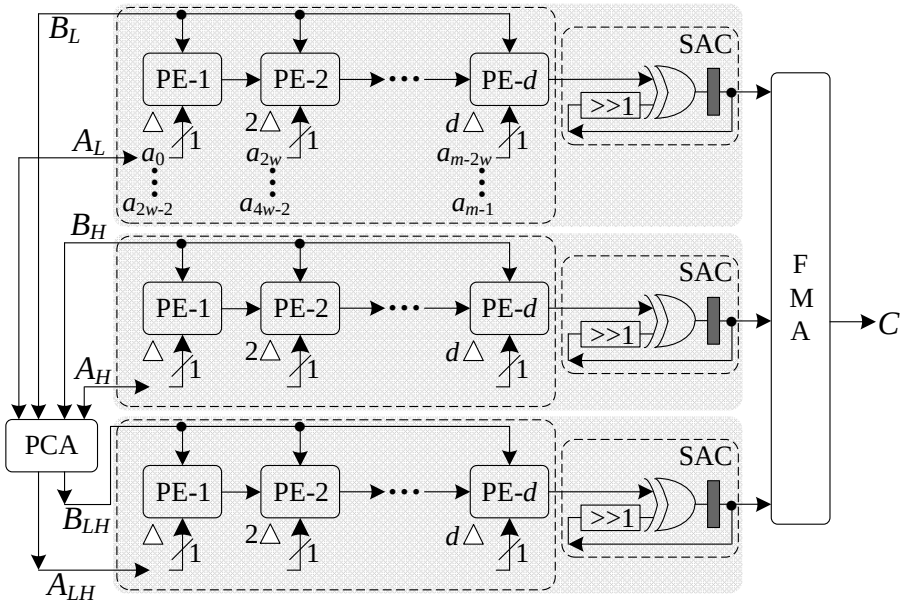

Figure 14: Proposed digit-serial systolic multiplier-III (DS-III), where Δ denotes unit delay and black block denotes register cell.

in every w cycles thereafter. Therefore, the proposed DS-III not only has efficient area complexity but also has high-throughput ability.

4.5 AREA AND TIME COMPLEXITIES

4.5.1 Complexities of BP-I and DS-I

BP-I requires w systolic arrays, where each array consists of d PEs. Each of regular PE has $(m + 3w)$ XOR gates (m XOR gates in XOR cell, and $3w$ XOR gates in M-II cell), m AND gates and $2m$ bit-registers. PRC cell requires $(3w - 3)$ XOR gates and $(m + 3w - 3)$ registers. PAT requires $(mw - w)$ XOR gates and equal number of registers. In total, BP-I has $(m^2 + 2mw - 3w^2 + 3w + md - d - 3)$ XOR gates, m^2 AND gates and $(2m^2 + 3w - 3)$ registers. BP-I has a latency of $(d + 1 + \log_2 w)$ cycles of duration of $2T_X$.

MBP-I has similar structure as BP-I, but it requires less registers and XOR gates. In total, MBP-I has $(m^2-m+3wd-3)$ XOR gates, m^2 AND gates and $(m^2+md+wm+2m-3d)$ registers. The latency and critical-path of MBP-I are the same as BP-I.

DS-I consists of $(d+1)$ PEs, where each of regular PE requires $(m+3w)$ XOR gates, m AND gates and m bit-registers, the same as that of BP-I. PE-0 requires 3 XOR gates (M-IV cell) and m bit-registers. While the AC cell contains m bit-registers and equal number of XOR gates. DS-I, in total thus requires $(md+3m-3w+3)$ XOR gates, dm AND gates and $(2dm+m)$ registers. After a latency of $(d+w)$ cycles, DS-I yields its desired output in every w cycles (duration of each cycle is $2T_X$).

4.5.2 Complexities of BP-II and DS-II

BP-II requires w systolic arrays, where each array consists of d PEs. Each of regular PE has $(m+3w)$ XOR gates (m XOR gates in XOR cell, and $3w$ XOR gates in M-VI cell), m AND gates and $(2m+2w-k1+k3)$ bit-registers. PRC cell requires $(2w^2-wk1+wk3+3w-3)$ XOR gates and $(mw+2w^2-wk1+wk3+2m+5w-k1+k3-3)$ registers. PAT requires $(mw-w)$ XOR gates and equal number of registers. In total, BP-II has $(3dw^2-w^2+m^2+2w-wk1+wk3-3)$ XOR gates, m^2 AND gates and $(2m^2+2dw^2-dwk1+dwk3+wm+2m+4w-k1+k3-3)$ registers. BP-II has a latency of $(d+3+\log_2 w)$ cycles of duration of (T_A+T_X).

MBP-II has similar structure as BP-II, but it requires less registers and XOR gates. In total, MBP-II has $(m^2-3w^2+5w+3wm-m-3-k1+k3)$ XOR gates, m^2 AND gates and $(m^2+md+wm+3wd-dk1+dk3+5w-k1+k3-4d+4m-3)$ registers. The latency and critical-path of MBP-II are the same as BP-II.

DS-II consists of $(d+1)$ PEs, where each of regular PE requires $(m+3w)$ XOR gates, m AND gates and $2m+2w-k1+k3$ bit-registers, the same as that of BP-II. PE-0 requires $(2w-k1+k3+3)$ XOR gates and $(m+2w-k1+k3)$ bit-registers. While the AC cell contains m bit-registers and equal number of XOR gates. DS-II, in total thus requires $(2m+3wd-w-k1+k3+3)$ XOR gates, dm AND gates and $(2dm+m+2wd-dk1+dk3)$ registers. After a latency of $(d+w+1)$ cycles, DS-II yields its desired output in every

Table 3: Comparison of area-time complexity of bit-parallel systolic multipliers

Design	AND	XOR	Register	Latency	Critical-path	ACT
[25]$*$	m^2	m^2+2m-1	$2m^2-2m$	mT_{cp}	T_A+T_X	T_{cp}
[56]	$2m^2+2m$	m^2+m	$3.5m^2+3.5m$	$(m+1)T_{cp}$	T_A+2T_X	T_{cp}
[57]	$2m^2$	$2m^2$	$7m^2$	$3mT_{cp}$	T_A+T_X	T_{cp}
[58]	$2m^2$	$2m^2$	$3m^2$	$(m+1)T_{cp}$	T_A+T_X	T_{cp}
[59]	m^2	$2m^2+m$	$2m^2$	mT_{cp}	T_A+2T_X	T_{cp}
[60]	$2m^2$	$2m^2$	$5m^2/2$	$(\lfloor m/2 \rfloor+2)T_{cp}$	T_A+T_X	T_{cp}
[61]^{*1}	m^2	γ^1	γ^2	γ^3	T_A+T_X	T_{cp}
BP-I	m^2	γ^4	$2m^2+3w-3$	$(d+1+\log_2 w)T_{cp}$	$2T_X$	T_{cp}
MBP-I	m^2	γ^5	γ^6	$(d+1+\log_2 w)T_{cp}$	$2T_X$	T_{cp}
BP-II	m^2	γ^7	γ^8	$(d+3+\log_2 w)T_{cp}$	T_A+T_X	T_{cp}
MBP-II	m^2	γ^9	γ^{10}	$(d+3+\log_2 w)T_{cp}$	T_A+T_X	T_{cp}
BP-III	$3m^2/4$	γ^{11}	γ^{12}	$(d+6+\log_2 w)T_{cp}$	T_A+T_X	T_{cp}

For BP-I/BP-II/MBP-I/MBP-II/DS-I/DS-II: $wd=m$; while for BP-III/DS-III: $wd=m/2$ T_{cp}: Time duration of critical-path. *: We have used XOR and AND gates to replace XNOR and NAND gates for this structure, just for a fair comparison. ^{1}: Here l=min$\{m-k1,\ k1-k2,\ k2-k3\}$.

$\gamma^1=m^2+2m+2lm+2l+2$ $\gamma^2=2m^2-2m-2lm-2l-2$ $\gamma^3=[m/(2l+2)+1+\log_2(2l+2)]T_{cp}$

$\gamma^4=m^2+2mw-3w^2+3w+md-d-3$ $\gamma^5=m^2-m+3wd-3$ $\gamma^6=m^2+md+wm+2m-3d$

$\gamma^7=3dw^2-w^2+m^2+2w-wk1+wk3-3$ $\gamma^8=2m^2+2dw^2-dwk1+dwk3+wm+2m+4w-k1+k3-3$ $\gamma^9=m^2-3w^2+5w+3wm-m-3-k1+k3$ $\gamma^{10}=m^2+md+wm+3wd-dk1+dk3+5w-k1+k3-4d+4m-3$ $\gamma^{11}=9m^2/8-3wm/4+6w+13m+7k1+7k2+7k3+9$ $\gamma^{12}=9m^2/8+3wm/4+6w+25m/2$

Table 4: Comparison of area-time complexity of digit-serial systolic multipliers

Design	AND	XOR	Register	Latency	Critical-path	ACT
[62]	γ^1	γ^2	$2m+d+t$	$(\lceil m/d \rceil + 1)T_{cp}$	γ^3	$(\lceil m/d \rceil + 1)T_{cp}$
[63][1]	$2md+m$	$2md$	γ^4	$3\lceil m/d \rceil T_{cp}$	γ^5	$3\lceil m/d \rceil T_{cp}$
DS-I	md	γ^6	$2dm+m$	$(d+w)T_{cp}$	$2T_X$	wT_{cp}
DS-II	md	γ^7	γ^8	$(d+1+w)T_{cp}$	T_A+T_X	wT_{cp}
DS-III	$3md/2$	γ^9	$9dm/4+67m/4$	$(d+6+w)T_{cp}$	T_A+T_X	wT_{cp}

For BP-I/BP-II/MBP-I/MBP-II/DS-I/DS-II: $wd=m$; while for BP-III/DS-III: $wd=m/2$ T_{cp}: Time duration of critical-path. [1]: $(s+1)$ refers to the pipelined stage in the PE, and $2m$ of MUX gates are not listed here.

$\gamma^1 = md + 2k1d - k1 + 2d - 1$

$\gamma^2 = md + 2k1d - k1 + d - 1$

$\gamma^3 = T_A + \lceil \log_2(d+1) \rceil T_X$

$\gamma^4 = \lceil m/d \rceil (10d + 1 + 9sd/2 + s)$

$\gamma^5 = d(T_A + T_X + T_{MUX})/(s+1)$

$\gamma^6 = md + 3m - 3w + 3$

$\gamma^7 = 2m + 3wd - w - k1 + k3 + 3$

$\gamma^8 = 2dm + m + 2wd - dk1 + dk3$

$\gamma^9 = 9md/4 + 63m/4 + 7k1 + 7k2 + 7k3 + 9$

w cycles (duration of each cycle is $(T_A + T_X)$). DS-II is regular and has high-throughput ability.

4.5.3 Complexities of BP-III and DS-III

BP-III requires three systolic-array modules, where each module has w systolic arrays (each array consists of d PEs). Each of regular PE has $(m/2 + vw)$ XOR gates $(1 \leq v \leq d-1)$, $m/2$ AND gates and $(m/2 + vw)$ bit-registers. PCA cell requires m XOR gates and m registers. SPAT requires $(2w + wm/2 - m/2)$ XOR gates and equal number of registers. FMA requires $(14m + 7k1 + 7k2 + 7k3 + 9)$ XOR gates and $27m/2$ registers. In total, BP-III has $(9m^2/8 - 3wm/4 + 6w + 13m + 7k1 + 7k2 + 7k3 + 9)$ XOR gates, $3m^2/4$ AND gates and $(9m^2/8 + 3wm/4 + 6w + 25m/2)$ registers. BP-III has a latency of $(d + 6 + \log_2 w)$ cycles of duration of $(T_A + T_X)$.

DS-III consists of $3d$ PEs, where each of regular PE requires $(m + vw)$ XOR gates, $m/2$ AND gates and $(m/2 + vw)$ bit-registers, the same as that of BP-III. While the SAC cell contains m bit-registers and equal number of XOR gates. PCA cell requires m XOR gates and m registers. FMA requires $(14m + 7k1 + 7k2 + 7k3 + 9)$ XOR gates and $27m/2$ registers. DS-III, in total thus requires $(9md/4 + 63m/4 + 7k1 + 7k2 + 7k3 + 9)$ XOR gates, $3dm/2$ AND gates and $(9dm/4 + 67m/4)$ registers. After a latency of $(d + 6 + w)$ cycles, DS-III yields its desired output in every w cycles (duration of each cycle is $(T_A + T_X)$).

4.5.4 Comparison of area and time complexities

The area complexity, in terms of logic gate count, register count; and time complexity in terms of latency, critical-path and ACT of proposed structures and existing structures of [25], [56-63] are listed in Tables 3 and 4, repsectively.

For bit-parallel systolic structures, proposed structures have the lowest latency and smaller area-complexity when compared with the existing designs. It is worth mentioning that this is the first time ever reported that proposed structures achieve flexible low-latency realization without any restriction. For digit-serial systolic structures, proposed structures have fixed critical-path, while the critical-paths of existing designs are a function of either filed-

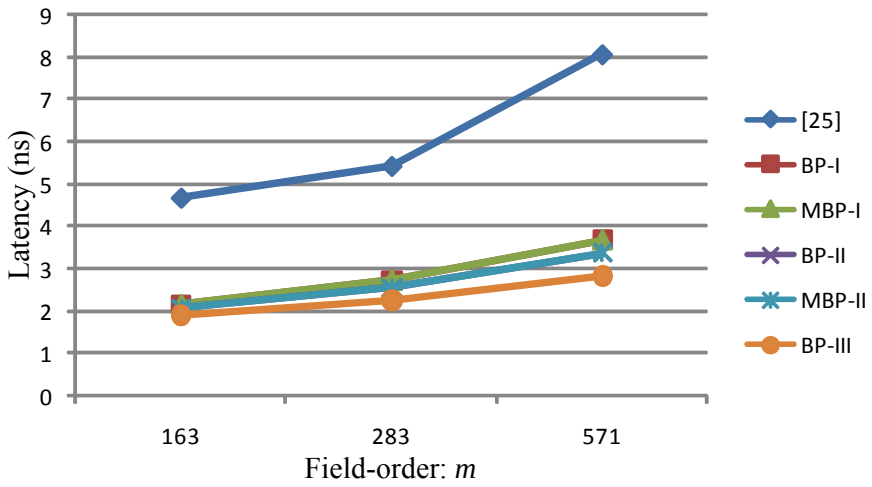

Figure 15: Comparison of latency of proposed and existing bit-parallel systolic multipliers.

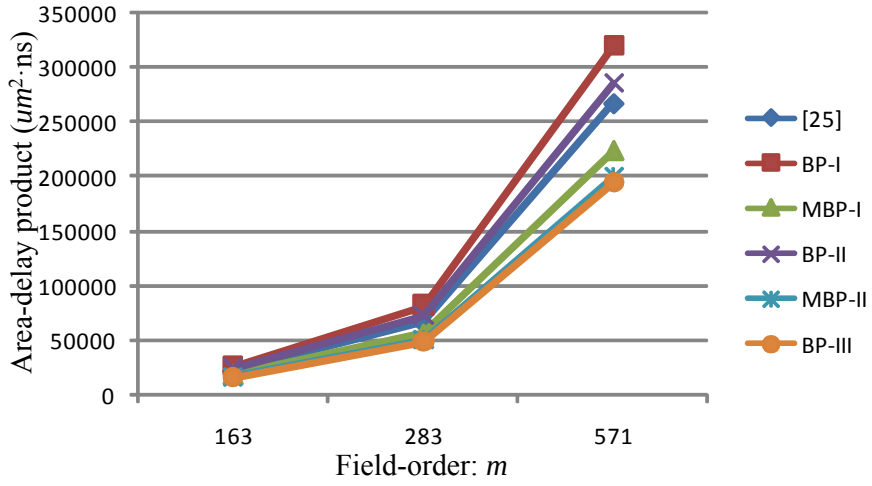

Figure 16: Comparison of area-delay products (ADP)s of proposed and existing bit-parallel systolic multipliers.

order or digit-size. The proposed structures have significant performance in time-complexity when compare to existing ones.

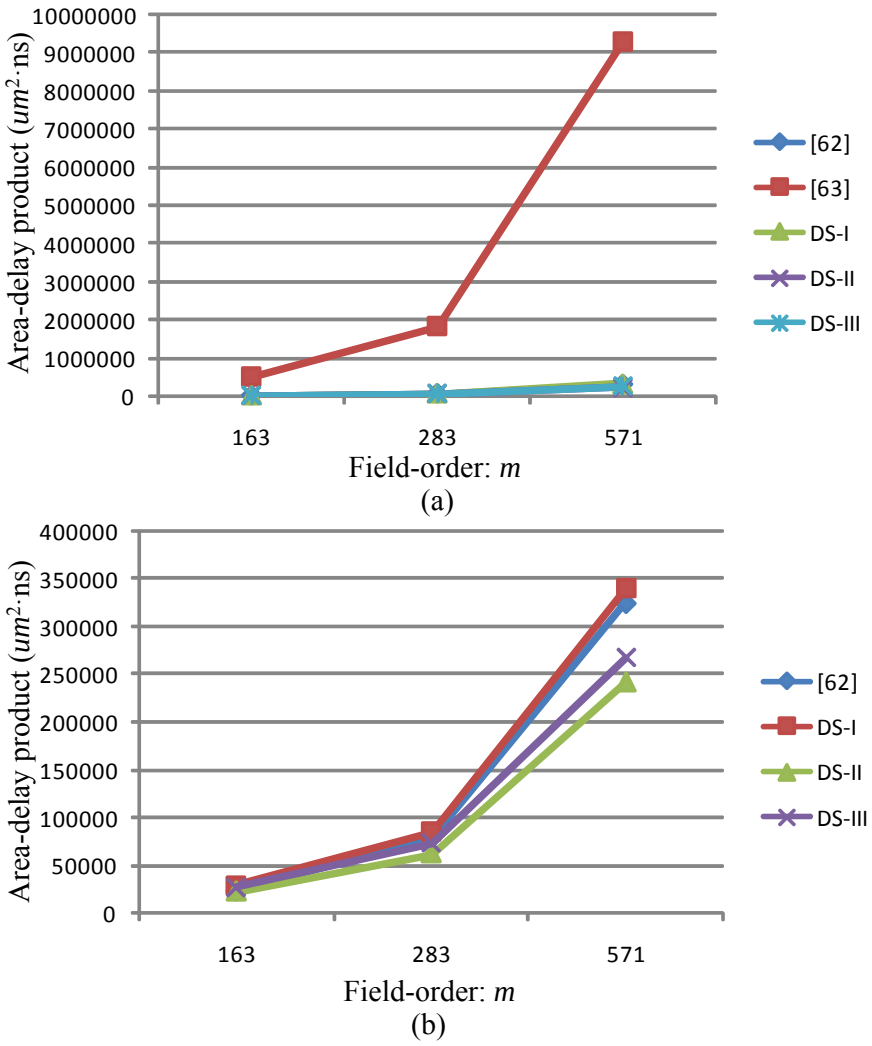

Figure 17: Comparison of area-delay products (ADP)s of proposed and existing digit-serial multipliers. (a) Comparison of ADPs of proposed and existing structures. (b) Detailed comparison.

4.5.5 ASIC implementation

We have also synthesized the proposed structures and the existing structures using NanGate's Library Creator by North Carolina State University's 45nm FreePDK [21] to obtain the area, time and power complexities of the designs for NIST pentanomials, i.e., $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$, $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$ and $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$. We have

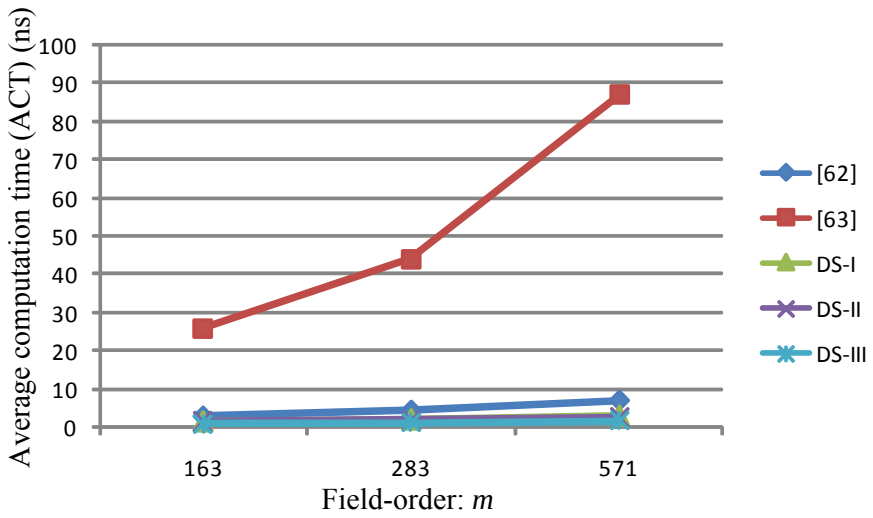

Figure 18: Comparison of average computation time (ACT) of proposed and existing digit-serial multipliers.

used $d = w = \sqrt{m}$ for BP-I/MBP-I/DS-I and BP-II/MBP-II/DS-II and all existing designs of [9-17] ($d = w = \sqrt{m/2}$ for BP-III/DS-III).

We have used those synthesis results to plot the latency and ADP for all bit-parallel designs as shown in Figs. 15 and 16, ADP and ACT for all digit-serial structures in Figs. 17 and 18, respectively, for three NIST pentanomials. As shown in Figs. 15 and 16, proposed bit-parallel structures significantly outperform the existing design, i.e., have lower latency and smaller ADP (MBP-I, MBP-II and BP-III). From Figs. 11 and 12, proposed digit-serial structures are found to involve significantly less ADP and ACT (the smaller the ACT, the higher the throughput rate) than the corresponding designs in [16-17]. Note that for a detailed comparison, Fig. 17(b) gives a more detailed comparison with [16] since Fig. 17(a) cannot give a clear comparison with [16]. It is noted that the PDP is found to be of similar trend as the ADP and found to be significantly less than those of competing designs. The proposed multipliers have significantly shorter latency, lower area-time complexity and higher throughput than the existing competing designs. The proposed designs can be used in various real application systems.

4.6 CONCLUSION

Low-latency high-throughput bit-parallel and digit-serial systolic structures for multipliers over $GF(2^m)$ based on NIST pentanomials are presented. We have proposed an algorithm where the multiplier is decomposed into several parallel processing arrays to lower the latency, and based on it we have suggested a pair of bit-parallel and digit-serial systolic multipliers. Another pair of bit-parallel and digit-serial structures are then presented based on a novel modular reduction operation, where the critical-paths are reduced to $(T_A + T_X)$ (throughput rate is increased). Data sharing strategy has been proposed to reduce area-complexity of the two pairs of systolic multipliers. The third pair of designs, KA-based bit-parallel and digit-serial multipliers, are proposed to enhance the throughput rate further. The synthesis results show that the proposed multipliers have significantly shorter latency, lower area-time complexity and higher throughput than the existing competing designs. The proposed designs, because of their flexibilities in latency choices, low area-time complexity and high throughput rate, can be used in various real application systems.

5.0 HIGH-THROUGHPUT FINITE FIELD MULTIPLIERS USING REDUNDANT BASIS FOR FPGA AND ASIC IMPLEMENTATIONS

In this chapter, we have proposed a novel recursive decomposition algorithm for RB multiplication to obtain high-throughput digit-serial implementation. Through efficient projection of SFG of the proposed algorithm, a highly regular processor-space flow-graph (PSFG) is derived. By identifying suitable cut-sets, we have modified the PSFG suitably and performed efficient feed-forward cut-set retiming to derive three novel multipliers which not only involve significantly less time-complexity than the existing ones but also require less area and less power consumption compared with the others. Both theoretical analysis and synthesis results confirm the efficiency of proposed multipliers over the existing ones. The synthesis results for FPGA and ASIC realization of the proposed designs and competing existing designs are compared. It is shown that the proposed high-throughput structures are the best among the corresponding designs, for FPGA and ASIC implementation. It is shown that the proposed designs can achieve up to 94% and 60% savings of ADPP on FPGA and ASIC implementation over the best of the existing designs, respectively.

5.1 INTRODUCTION

Finite field multiplication over $GF(2^m)$ is a basic operation frequently encountered in modern cryptographic systems such as the ECC and error control coding [1-3]. Moreover, multiplication over a finite field can be used further to perform other field operations, e.g., division, exponentiation, and inversion [4-6]. Multiplication over $GF(2^m)$ can be implemented on a general purpose machine, but it is expensive to use a general purpose machine to implement

cryptographic systems in cost-sensitive consumer products. Besides, a low-end microprocessor cannot meet the real-time requirement of different applications since word-length of these processors is too small compared with the order of typical finite fields used in cryptographic systems. Most of the real-time applications, therefore, need hardware implementation of finite field arithmetic operations for the benefits like low-cost and high-throughput rate.

The choice of basis to represent field elements, namely the polynomial basis, normal basis, triangular basis and RB has a major impact on the performance of the arithmetic circuits [7-9]. The multipliers based on RB [6, 10] have gained significant attention in recent years due to their several advantages. Not only do they offer free squaring, as normal basis does, but also involve lower computational complexity and can be implemented in highly regular computing structures [13-14].

Several digit-level serial/parallel structures for RB multiplier over $GF(2^m)$ have been reported in the last years [13-14] after its introduction by Wu *et al.* [68]. An efficient serial/parallel multiplier using redundant representation has been presented in [70]. A bit-serial word-parallel (BSWP) architecture for RB multiplier has been reported by Namin *et. al* [71]. Several other RB multipliers also have been developed by the same authors in [13-14] for reducing the complexity of implementation and for high-speed realization. We find that the hardware utilization efficiency and throughput of existing structures of [13-14] can be improved by efficient design of algorithm and architecture.

In this chapter, we aim at presenting efficient digit-level serial/parallel designs for high-throughput finite field multiplication over $GF(2^m)$ based on RB. We have proposed an efficient recursive decomposition scheme for digit-level RB multiplication, and based on that we have derived parallel algorithms for high throughput digit-serial multiplication. We have mapped the algorithm to three different high-speed architectures by mapping the parallel algorithm to a regular 2-dimensional SFG array, followed by suitable projection of SFG to 1-dimensional PSFG, and the choice of feed-forward cut-set to enhance the throughput rate. Our proposed digit-serial multipliers involve significantly less area-time-power complexities than the corresponding existing designs. FPGA has evolved as a mainstream dedicated computing platform. FPGAs however do not have abundant number of registers to be used in the multiplier. Therefore, we have modified the proposed algorithm and architecture

for reduction of register-complexity particularly for the implementation of RB multipliers on FPGA platform. Apart from these we also present a low critical-path digit-serial RB multiplier for very high throughput applications.

The rest of this chapter is organized as follows: The proposed algorithm for finite field RB multiplication over $GF(2^m)$ is presented in Section 5.2. The proposed structures for high-throughput digit-serial realization of the multiplications are derived from the proposed algorithm in Section 5.3. In Section 5.4, we have discussed the estimation of hardware and time complexities along with the comparative performance of proposed designs over the recent competing designs. Conclusions are presented in Section 5.5.

5.2 MATHEMATICAL FORMULATION

5.2.1 Brief review of existing digit-serial RB multiplier

Assuming x to be a primitive nth root of unity, elements in finite field $GF(2^m)$ can be represented in the form:

$$A = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \tag{5.1}$$

where $a_i \in GF(2)$, for $0 \leq i \leq n-1$, such that the set $\{1,\ x,\ x^2, \ldots, x^{n-1}\}$ is defined as the RB for finite field elements, where n is a positive integer not less than m [68], [70].

For a finite field $GF(2^m)$, when $(m+1)$ is prime and 2 is a primitive root modulo $(m+1)$, there exists a type I ONB [68], where x is an element of $GF(2^m)$, and $n = m + 1$.

Let $A,\ B \in GF(2^m)$ be expressed in RB representation as

$$A = \sum_{i=0}^{n-1} a_i x^i \tag{5.2}$$

$$B = \sum_{i=0}^{n-1} b_i x^i \tag{5.3}$$

where $a_i,\ b_i \in GF(2)$. Let C be the product of A and B, which can be expressed as following equations

$$\begin{aligned} C = A \cdot B &= \sum_{i=0}^{n-1} (x^i b_i) \cdot A \\ &= \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} b_i x^{(i+j)}) a_j \\ &= \sum_{j=0}^{n-1} (\sum_{i=0}^{n-1} b_{(i-j)_n} x^i) a_j \\ &= \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} b_{(i-j)_n} a_j) x^i \end{aligned} \tag{5.4}$$

where $(i-j)_n$ denotes modulo n reduction. Define $C = \sum_{i=0}^{n-1} c_i x^i$, where $c_i \in GF(2)$, we have

$$c_i = \sum_{i=0}^{n-1} b_{(i-j)_n} a_j. \tag{5.5}$$

In the recently proposed RB multipliers of [13] and [14], both operands A and B are decomposed into a number of blocks to achieve digit-serial multiplication, and after that the partial products corresponding to these blocks are added together to obtain the desired product word. The existing digit-serial RB multiplication algorithm is stated as follows:

Existing Algorithm Existing digit-serial multiplication algorithm [13] and [14]

Inputs: $A = (\underbrace{a_0 \dots a_{w-1}}_{A'_0} \underbrace{a_w \dots a_{2w-1}}_{A'_1} \dots \underbrace{a_{(t-1)w} \dots a_{n-1}}_{A'_{t-1}})$
$= (A'_0, \dots, A'_{t-1})$,
and $B = \underbrace{b_0 \dots b_{w-1}}_{B'_0} \underbrace{b_w \dots b_{2w-1}}_{B'_1} \dots \underbrace{b_{(t-1)w} \dots b_{n-1}}_{B'_{t-1}}$
$= (B'_0, \dots, B'_{t-1})$
are two RB representation elements, for $t = \lceil n/w \rceil$.

Output: $C = A \cdot B = (c_0, \dots, c_{n-1})$

1. Initialization: $t = \lceil n/w \rceil,\ r_{e,i}^{(-1)} = 0$ for $e = 0, \dots, t-1$ and $i = 0, \dots, n-1$

2. For all values of $i = 0, 1, 2, \dots, n-1$, compute

3. For all values of $e = 0, 1, 2, \ldots, t-1$, compute

4. For $g = 0$ to $w - 1$, compute

5. $r_{e,i}^{(g)} = r_{e,i}^{(g-1)} + a_{ew+g} b_{(i-ew-g)n}$

6. End For

7. End For

8. $c_i = \sum_{e=0}^{t-1} r_{e,i}^{(w-1)}$

9. End For

Step 5 of existing algorithm refers to the computation of digit-wise partial products for the digit-serial multiplication where operands A and B are decomposed into a number of digits, and step 8 refers to the addition of those partial products to compute the product word.

Although the existing algorithm of [13] and [14] is the most efficient one out of all reported algorithms for digit-serial multiplication, we find that the hardware utilization efficiency and throughput of existing structures of [13]-[14] could be improved further by efficient design of algorithm and architecture. Particularly, due to its larger number of digit-wise partial products (step 5), and extra time for addition of those partial products (step 8), which not only increases the ACT to perform the multiplication but also involves extra hardware resources for storage and addition of larger number of partial products.

5.2.2 Proposed digit-serial RB multiplication algorithm

Alternatively, we can write (5.5) into a bit-level matrix-vector form as the following steps:

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 & b_{n-1} & \cdots & b_1 \\ b_1 & b_0 & \cdots & b_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}. \tag{5.6}$$

Looking at the structure of bit matrix in (5.6), we can define n bit-shifted (reduced) forms of operand B as follows

$$\begin{aligned} B^0 &= \sum_{i=0}^{n-1} b_i^0 x^i = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1} \\ B^1 &= \sum_{i=0}^{n-1} b_i^1 x^i = b_{n-1} + b_0 x + \cdots + b_{n-2} x^{n-1} \\ &\cdots \quad \cdots \quad \cdots \\ B^{n-1} &= \sum_{i=0}^{n-1} b_i^{n-1} x^i = b_1 + b_2 x + \cdots + b_0 x^{n-1} \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} b_0^{i+1} &= b_{n-1}^i \\ b_j^{i+1} &= b_{j-1}^i, \text{ for } 1 \le j \le n-2. \end{aligned} \tag{5.8}$$

The recursions on (5.8) can be extended further to have

$$b_j^{i+s} = \begin{cases} b_{n-s+j}^i, & \text{for } 0 \le j \le s-1 \\ b_{j-s}^i, & \text{otherwise} \end{cases} \tag{5.9}$$

where $1 \le s \le n-1$.

Let Q and P be two integers such that $n = QP + r$, where $0 \le r < P$. For simplicity of discussion, we assume[1] $r = 0$, and decompose the input operand A into Q number of bit-vectors A_u for $u = 0, 1, \cdots, Q-1$, as follows:

$$\begin{aligned} A_0 &= [a_0 \ a_Q \cdots \ a_{n-Q}] \\ A_1 &= [a_1 \ a_{Q+1} \cdots a_{n-Q+1}] \\ &\cdots \quad \cdots \quad \cdots \\ A_{Q-1} &= [a_{Q-1} \ a_{2Q-1} \ \cdots \ a_{n-1}]. \end{aligned} \tag{5.10}$$

[1]When $r \neq 0$, we can append $(Q-r)$ number of zeros to each of the operands to satisfy the condition $n = QP$.

Similarly, we can generate Q number of shifted operand vectors B_u for $u = 0, 1, \cdots, Q-1$, as follows:

$$\begin{aligned} B_0 &= [B^0 \ B^Q \cdots \ B^{n-Q}] \\ B_1 &= [B^1 \ B^{Q+1} \cdots \ B^{n-Q+1}] \\ &\cdots \quad \cdots \quad \cdots \\ B_{Q-1} &= [B^{Q-1} \ B^{2Q-1} \cdots \ B^{n-1}]. \end{aligned} \tag{5.11}$$

The product $C = AB$ given by the bit-level matrix-vector product in (5.6) can be decomposed into Q inner-products of vectors A_u and B_u for $u = 0, 1, \cdots, Q-1$ as:

$$\begin{aligned} C = AB &= B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T \\ &= \sum_{u=0}^{Q-1} B_u A_u^T = \sum_{u=0}^{Q-1} \overline{C_u} \end{aligned} \tag{5.12}$$

where $\overline{C_u}$ denotes

$$\overline{C_u} = B_u A_u^T. \tag{5.13}$$

Note that each A_u for $u = 0, 1, \cdots, Q-1$ is a P-point bit-vector and each B_u for $u = 0, 1, \cdots, Q-1$ is P bit-shifted forms of operand B. From (5.12) and (5.13) we can find that the desired multiplication can be performed by Q cycles of successive accumulation of $\overline{C_u}$ for $u = 0, 1, \cdots, Q-1$, while each $\overline{C_u}$ can be computed as $\overline{C_u} = \sum_{v=0}^{P-1} B^{u+vQ} a_{u+vQ}$. The proposed digit-serial multiplication algorithm based on (5.12) and (5.13) is described in Algorithm 5.1.

Algorithm 5.1 Proposed digit-serial multiplication algorithm

Inputs: A and B are the pair of elements (in RB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 $D = 0$

2. Multiplication step

   2.1. for $u = 0$ to $Q - 1$

   2.2. for $v = 0$ to $P - 1$

   2.3. $D = D + B_u A_u^T$

   2.4. end for

   2.5. end for

3. Final step

   3.1. $C = D$

---

where step 2.3 refers to the digit-serial multiplication process. According to our proposed algorithm, we generate less number of partial products, and partial products are accumulated as soon as they are computed, which not only shortens the ACT, but also significantly reduces register and adder complexities of proposed structures over that of existing ones in [13] and [14].

5.3 DERIVATION OF PROPOSED HIGH-THROUGHPUT STRUCTURES FOR RB MULTIPLIERS

In this section, we derive the proposed multipliers from the SFG of the proposed Algorithm 5.1.

5.3.1 Proposed structure-I

According to (5.12) and (5.13), the RB multiplication can be represented by the 2-dimensional SFG (shown in Fig. 19) consisting of Q parallel arrays, where each array consists of $(P-1)$ bit-shifting nodes (S node), P multiplication nodes (M nodes) and $(P-1)$ addition nodes (A nodes). There are two types of S nodes (S-I node and S-II node). Function of S nodes is depicted in Fig. 19(b), where S-I node performs circular bit-shifting by one position and

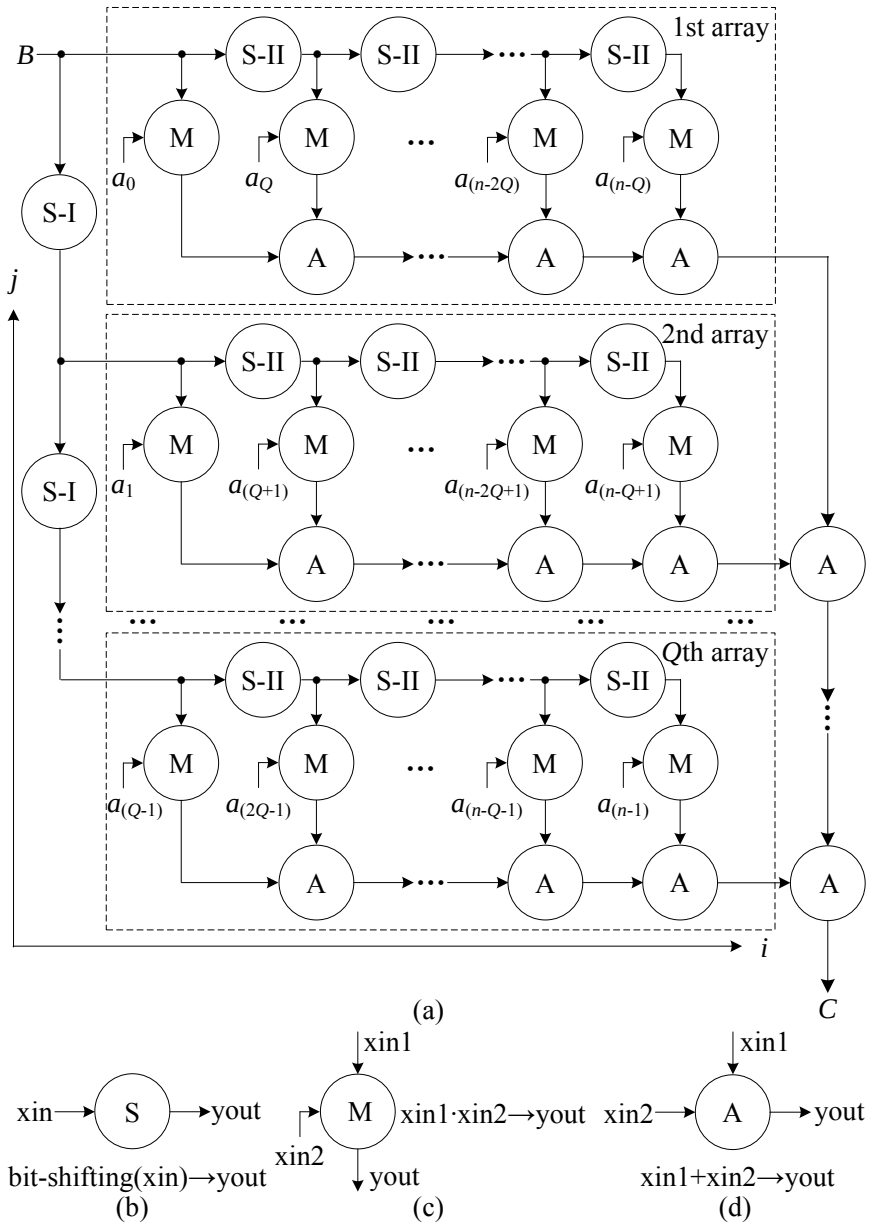

Figure 19: Signal-flow graph (SFG) for parallel realization of RB multiplication over $GF(2^m)$ based on (12) and (13). (a) The proposed SFG. (b) Functional description of S node, where S-I node performs circular bit-shifting of one position and S-II node performs circular bit-shifting by Q positions. (c) Functional description of M node. (d) Functional description of A node.

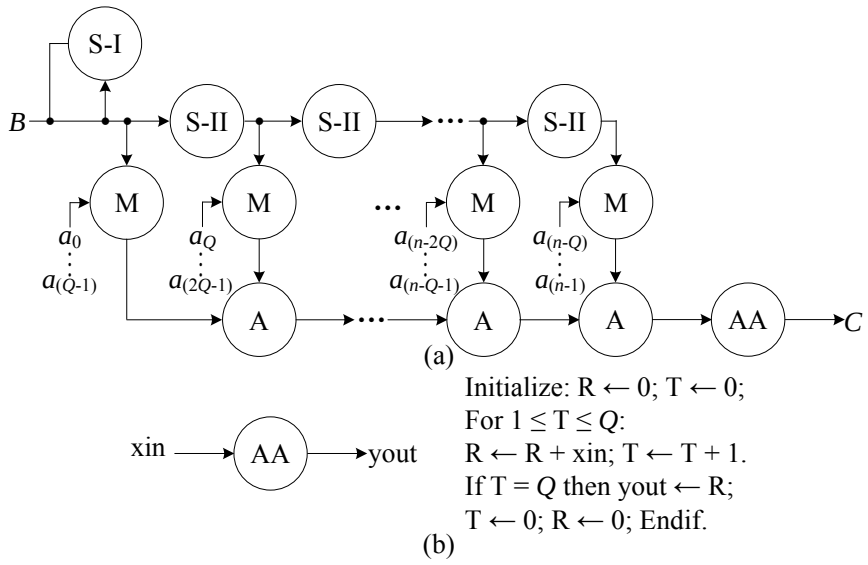

Figure 20: Processor-space flow graph (PSFG) of digit-serial realization of finite field RB multiplication over $GF(2^m)$. (a) The proposed PSFG. (b) Functional description of add-accumulation (AA) node.

S-II node performs circular bit-shifting by Q positions for the degree reduction requirement. Functions of M nodes and A nodes are depicted in Fig. 19(c) and 19(d), respectively. Each of the M nodes performs an AND operation of a bit of serial-input operand A with bit-shifted form of operand B, while each of the A nodes performs an XOR operation. The final addition of the output of Q arrays of Fig. 1 can be performed by bit-by-bit XOR of the operands in $(Q-1)$ number of A nodes as depicted in Fig. 19. The desired product word is obtained after the addition of Q parallel output of the arrays.

For digit-serial realization of RB multiplier, the SFG of Fig. 19 can be projected along j-direction to obtain a PSFG as shown in Fig. 20, where P input bits are loaded in parallel to multiplication nodes during each cycle period. The functions of nodes of PSFG are the same as those of corresponding nodes in the SFG of Fig. 19 except an extra add-accumulation (AA) node. The function of the AA node is, as described in Fig. 20(b), to execute the accumulation operation for Q cycles to yield the desired result thereafter. For efficient realization of a digit-serial RB multiplier, we can perform feed-forward cut-set retiming in a

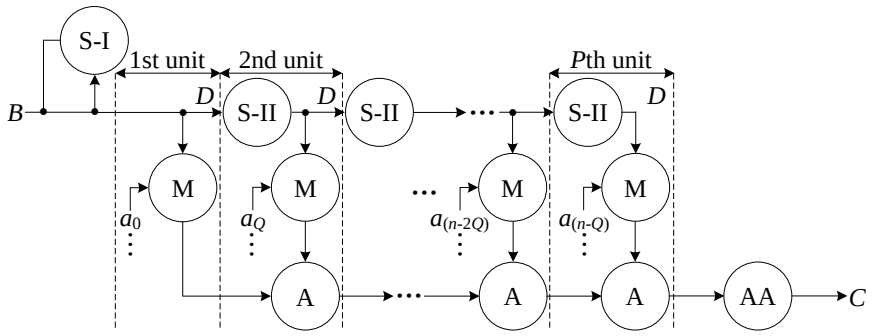

Figure 21: Cut-set retiming of PSFG of finite field RB multiplication over $GF(2^m)$, where "D" denotes delay.

regular interval in the PSFG as shown in Fig. 21. As a result of cut-set retiming of the Fig. 21, the minimum duration of each clock period of PSFG is reduced to $(T_A + T_X)$.

The PSFG of Fig. 21, is mapped to the high-throughput digit-serial RB multiplier (shown in Fig. 22), referred to as proposed structure-I (PS-I). PS-I contains three modules, namely the bit-permutation module (BPM), partial product generation module (PPGM) and finite field accumulator module. The BPM of Fig. 22 performs rewiring of bits of operand B to feed its output to P partial product generation units (PPGU)s according to the S nodes of PSFG of Fig. 21, as shown in Fig. 22(b). The AND cell, XOR cell and register cell of PPGM perform the function of M node, A node and delay imposed by the retiming of PSFG of Fig. 21, respectively. Structures and functions of AND cell, XOR cell and register cell are shown in Fig. 22(c), (d) and (e), respectively. The input operands are fed to PPGU in staggered manner to meet the timing requirement in systolic pipeline. The accumulator consists of n parallel bit-level accumulation cells (as shown in Fig. 22(f)). The newly received input is then added with the previously accumulated result and the result is stored in the register cell to be used during the next cycle. The duration of minimum cycle period of the PS-I is $(T_A + T_X)$. The proposed digit-serial design gives the first output of desired product $(P + Q)$ cycles after the pair of operands are fed to the structure, while the successive output are produced at the interval of Q cycles thereafter.

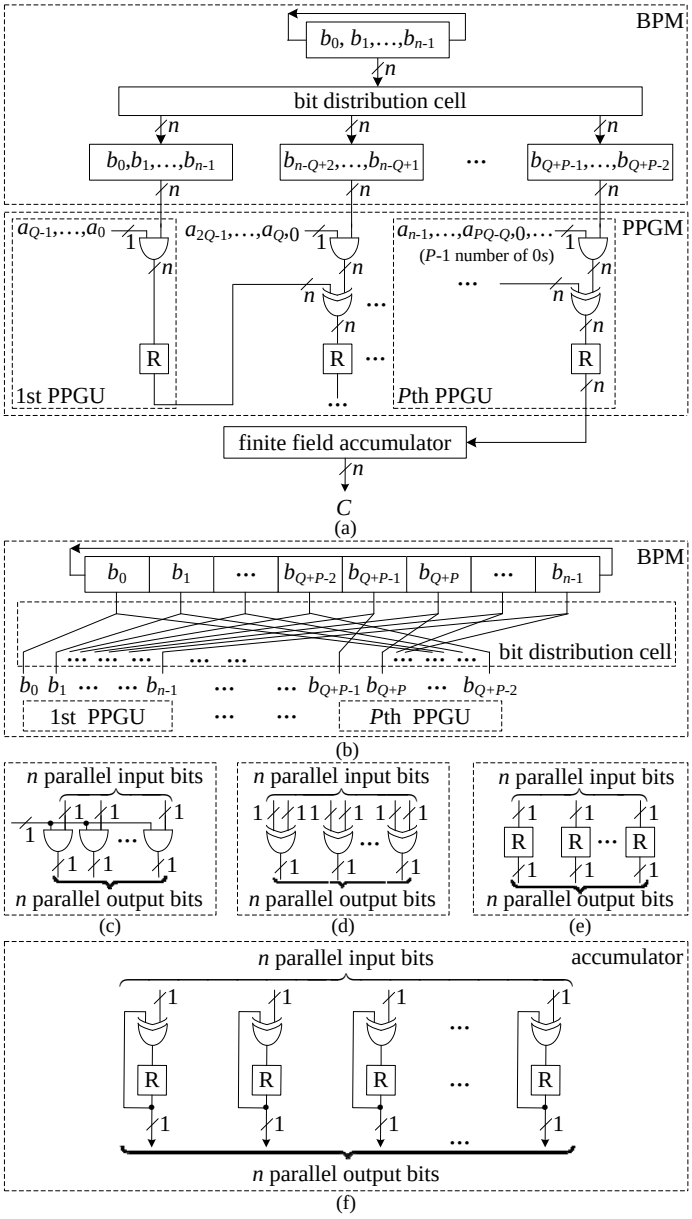

Figure 22: Proposed structure-I (PS-I) for RB multiplier, where "R" denotes a register cell. (a) Detailed structure of the RB multiplier. (b) Structure of the bit-permutation module (BPM). (c) Structure of the AND cell in the partial product generation module (PPGM). (d) Structure of the XOR cell in the PPGM. (e) Structure of the register cell in the PPGM. (f) Structure of the finite field accumulator.

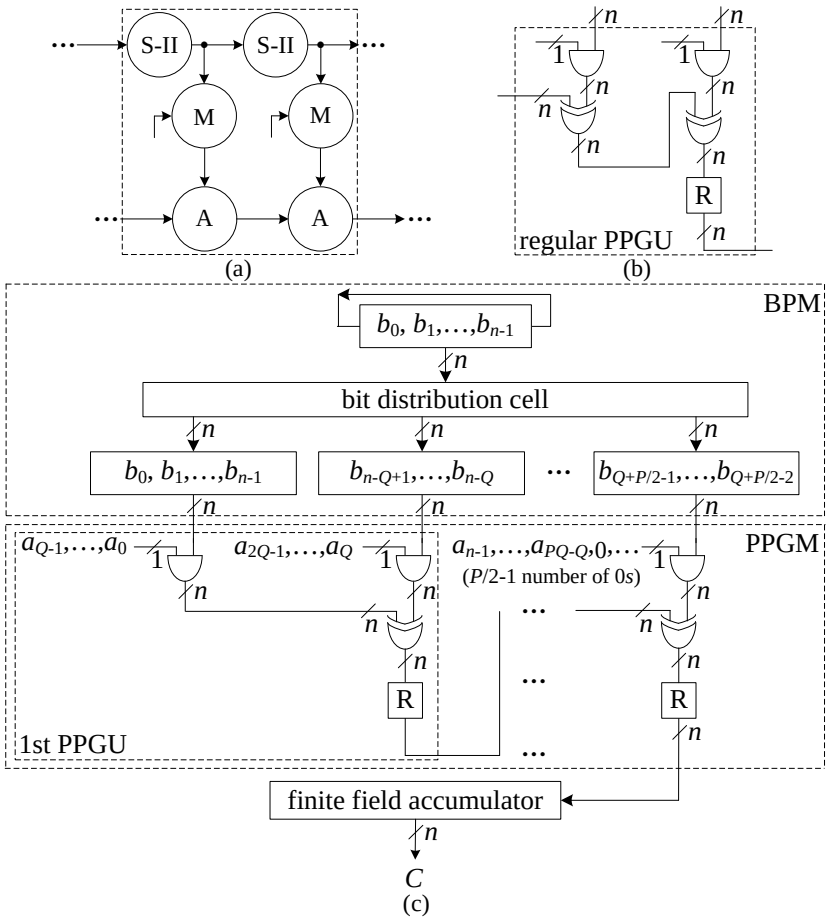

Figure 23: PS-I for RB multiplier when $d = 2$. (a) Proposed cut-set retiming of PSFG when $d = 2$. (b) Detailed internal structure of merged regular PPGU. (c) Corresponding PS-I for the case $d = 2$.

5.3.2 Modification of proposed structure-I

For any integer value of P, we can have $(P = kd + l)$, where $0 \leq l < d$ and $d < P$. Without loss of generality, for simplicity of discussion, we can assume $l = 0$. The approach proposed here for $l = 0$ however can be easily extended to the cases where $l \neq 0$. Define $0 \leq h \leq k-1$, and $0 \leq f \leq d-1$, such that (5.13) can be rewritten as

$$\overline{C_u} = \sum_{h=0}^{k-1} \sum_{f=0}^{d-1} B^{u+fhQ} a_{u+fhQ} \tag{5.14}$$

Based on (5.14), we can modify the retiming of PSFG of Fig. 21 to derive suitable digit-level architecture for RB multiplier over $GF(2^m)$. For example, to obtain the proposed structure for $d = 2$, a pair of S nodes, a pair of M nodes and a pair of A nodes of the PSFG of Fig. 21 can be merged to form a macro-node as shown within the dashed-lines in Fig. 23. Each of these macro-nodes can be implemented by a new PPGU to obtain a PPGM of $P/2$ PPGUs. Accordingly, two regular PPGUs in the structure of Fig. 22 can be emerged into a new regular PPGU as shown in Fig. 23(b), which consists of two AND cells and two XOR cells (the first PPGU requires only one XOR cell). The functions of AND cell, XOR cell and register cell are the same as those described in Fig. 4. The critical path of the structure of Fig. 23(c) amounts to $(T_A + 2T_X)$. The first output of desired product is available from this structure after a latency of $(P/2+Q)$ cycles, while the successive outputs are available thereafter in each Q cycles of duration $(T_A + 2T_X)$. The technique used to derive the structure for $d = 2$ may be extended for any value of d, to obtain a structure consisting of (P/d) PPGUs.

The technique based on (5.14) can significantly reduce the register complexity of the proposed structure, since P consecutive PPGUs of the PS-I can be merged together to form $(k = P/d)$ units to be processed concurrently. This strategy is quite useful for FPGA-based implementation since the value of d can be chosen appropriately, such that the PSFG nodes selected to be processed in a cycle can be mapped to a basic unit of FPGA with low register complexity.

5.3.3 Proposed structure-II

We can further transform the PSFG of Fig. 21 to reduce the latency and hardware complexity of PS-I. To obtain the proposed structure, $(P-1)$ serially-connected A nodes of the PSFG of Fig. 21 are merged into a pipeline form of $(P-2)$ A nodes as shown within the dashed-box in Fig. 24(a). These pipelined A nodes can be implemented by a pipelined XOR tree, as shown in Fig. 24(b). Since all the AND cells can be processed in parallel, there is no need of using extra "0"s on the input path to meet the timing requirement in systolic pipeline. The critical path and throughput of PS-II are the same as those of PS-I. Similarly, PS-II can

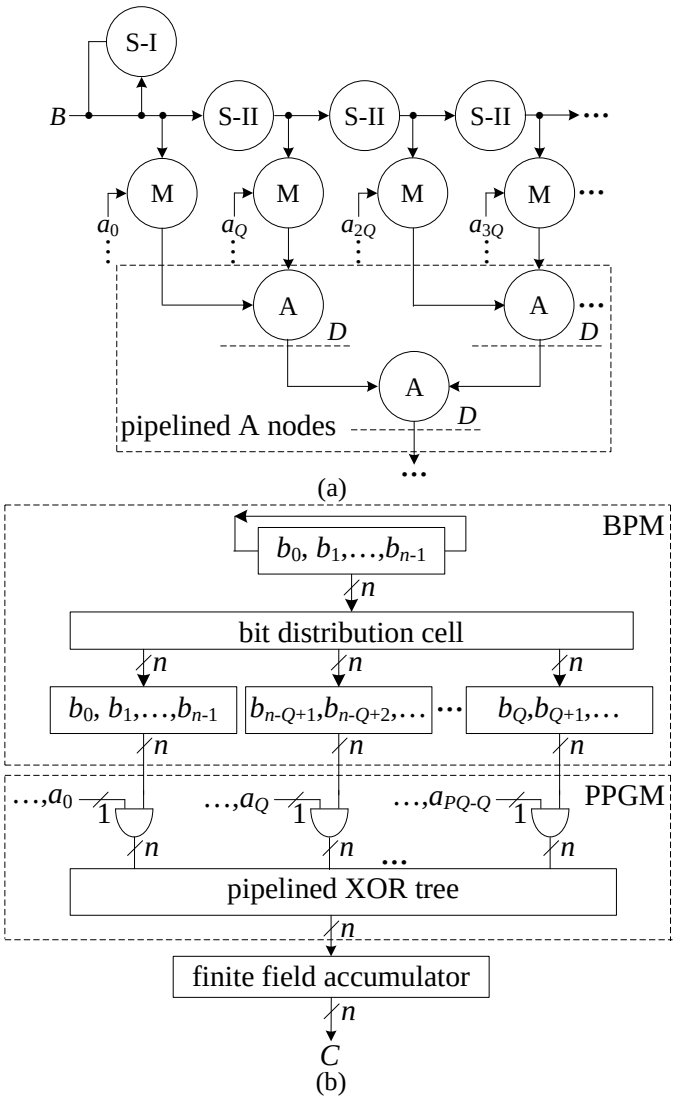

Figure 24: Proposed structure-II (PS-II) for RB multiplier, where "R" denotes a register cell. (a) Modified PSFG. (b) Structure of RB multiplier.

be easily extended to larger values of d to have low register-complexity structures to achieve lower hardware complexity implementation.

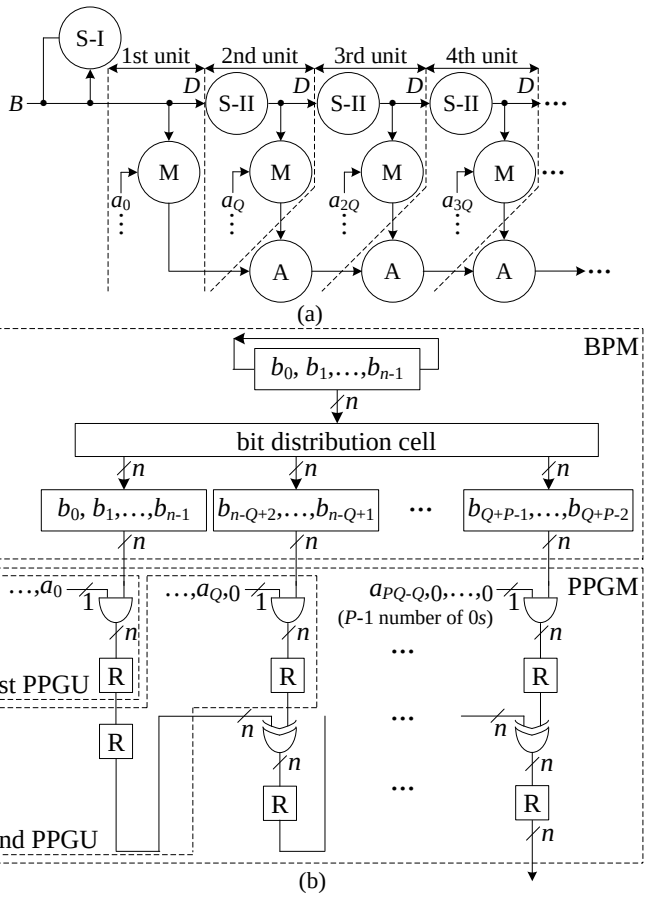

Figure 25: Novel cut-set retiming of PSFG and its corresponding structure: PS-III. (a) Cut-set retiming. (b) BPM and PPGM of PS-III.

5.3.4 Proposed structure-III

Since the S nodes of Fig. 21 perform only the bit-shifting operations they do not involve any time consumption. Therefore, we can introduce a novel cut-set retiming to reduce the critical-path further, as shown in Fig. 25(a). It can be observed that the cut-set retiming allows to perform the bit-addition and bit-multiplication concurrently, so that the critical-path is reduced to $\max\{T_A,\ T_X\}=T_X$, i.e., the throughput of the design is increased. The proposed high-throughput structure (PS-III) of RB multiplier thus derived is shown in Fig. 25(b). It consists of $(P+1)$ PPGUs, and each PPGU consists of one AND cell, one XOR

cell and two register cells. The proposed structure yields the first output of desired result $(P + Q + 1)$ cycles after the first input is fed to the structure, while the successive outputs are available in each Q cycles.

5.4 AREA-TIME-POWER COMPLEXITIES

5.4.1 Complexities of PS-I, PS-II and PS-III

PS-I requires P PPGUs, where each of the $(P-1)$ regular PPGUs consists of n XOR gates and n AND gates. The finite field accumulator requires n XOR gates and n bit-registers. The proposed design in total requires Pn XOR gates, Pn AND gates and $(Pn + 2n)$ bit-registers. After a latency of $(P + Q)$ cycles, PS-I gives the desired output word in every Q cycles of duration $(T_A + T_X)$.

PS-I for any value of d consists of (P/d) PPGUs. The complete structure of the multiplier thus requires Pn XOR gates, Pn AND gates and $(Pn/d + 2n)$ bit-registers. The latency of the structure amounts to $(P/d + Q)$ cycles, where the duration of minimum cycle period is $\{T_A + (1 + \lceil \log_2 d \rceil)T_X\}$.

PS-II has similar area-time complexities as those of PS-I except that it involves less registers and lower latency than the latter. In total, PS-II requires $(Pn + n)$ registers and yields its first result after a latency of $(\log_2 P + Q)$ cycles. For any value of d, PS-II requires $\{Pn/d + n\}$ registers.

PS-III requires $(P + 1)$ PPGUs, where each of the $(P - 2)$ regular PPGUs consists of n XOR gates, n AND gates and $2n$ bit-registers. The proposed design in total would require Pn XOR gates and Pn AND gates. Besides, it needs a total of $(2Pn + 2n)$ bit-registers. After a latency of $(P + Q + 1)$ cycles, PS-III gives the desired output word in every Q cycles of duration T_X.

Table 5: Comparison of area-time complexities of digit-serial RB multipliers

Design	AND	XOR	Register	Latency	Critical-path	ACT
[68]	Pn	$P(n-1)$	n	QT_{cp}	γ^1	QT_{cp}
[70]	Pn	Pn	$2n$	QT_{cp}	γ^2	QT_{cp}
[71]	Pn	$Pn+n$	n	QT_{cp}	γ^2	QT_{cp}
[13]	Pn	$(2P-1)n$	$(P+1)n$	$QT_{cp}+\lceil\log_2 P\rceil T_X$	T_A+T_X	γ^3
[14]	γ^4	$(2P-1)n$	$Pn+\lceil\log_2 Q\rceil$	$QT_{cp}+\lceil\log_2 P\rceil T_X$	T_A+T_X	γ^3
PS-I[1]	Pn	Pn	$Pn+2n$	$(P+Q)T_{cp}$	T_A+T_X	QT_{cp}
PS-II[1]	Pn	Pn	$Pn+n$	$(\lceil\log_2 P\rceil+Q)T_{cp}$	T_A+T_X	QT_{cp}
PS-I[2]	Pn	Pn	$Pn/d+2n$	$(P/d+Q)T_{cp}$	γ^5	QT_{cp}
PS-II[2]	Pn	Pn	$Pn/d+n$	$(\lceil\log_2 P\rceil+Q)T_{cp}$	γ^5	QT_{cp}
PS-III	Pn	Pn	$2Pn+2n$	$(P+Q+1)T_{cp}$	T_X	QT_{cp}

T_{cp}: Time duration of critical-path.

d: d is the number of bits of operand A fed to each PPGU during each cycle period.

[1]: refers to the structure with $d=1$.

[2]: refers to the structure with $1<d<P$.

$\gamma^1 = T_A + \lceil\log_2 n\rceil T_X$

$\gamma^2 = T_A + \lceil\log_2 (P+1)\rceil T_X$

$\gamma^3 = QT_{cp} + \lceil\log_2 P\rceil T_X$

$\gamma^4 = Pn + \lceil\log_2 Q\rceil$

$\gamma^5 = T_A + (1+\lceil\log_2 d\rceil) T_X$

Table 6: Comparison of area-time complexities of different multipliers where there exist a type I ONB ($n = m + 1$)

Design	AND	XOR	Register	Latency	CP	ACT
[69]ONBI	$P(2m-1)$	$P(2m-2)$	$2m$	QT_{cp}	τ^1	QT_{cp}
[74]ONBI	Pm	$P(2m-2)$	$2m$	QT_{cp}	τ^1	QT_{cp}
[75]ONBI^0	$Pm/2 + m/2$	τ^2	$2m$	QT_{cp}	τ^1	QT_{cp}
[75]ONBI^1	$Pm - P + m$	τ^3	$2m$	QT_{cp}	τ^1	QT_{cp}
[77]ONBI^2	τ^4	τ^5	$3m$	QT_{cp}	τ^6	QT_{cp}
[76]ONBI^3	τ^7	τ^3	$3m$	QT_{cp}	τ^8	QT_{cp}
[77]ONBI^4	$Pm + P$	$2Pm + P$	$3m$	QT_{cp}	τ^9	QT_{cp}
PS-I$^!$	$Pm + P$	$Pm + P$	$(P+2)(m+1)$	$(P+Q)T_{cp}$	τ^{10}	QT_{cp}
PS-II$^!$	$Pm + P$	$Pm + P$	$(P+1)(m+1)$	$(\lceil \log_2 P \rceil + Q)T_{cp}$	τ^{10}	QT_{cp}
PS-I*	$Pm + P$	$Pm + P$	τ^{11}	$(P/d + Q)T_{cp}$	τ^{12}	QT_{cp}
PS-II*	$Pm + P$	$Pm + P$	τ^{13}	$(\lceil \log_2 P \rceil + Q)T_{cp}$	τ^{12}	QT_{cp}
PS-III	$Pm + P$	$Pm + P$	$2(P+1)(m+1)$	$(P+Q+1)T_{cp}$	T_X	QT_{cp}

CP: Critical-path. T_{cp}: Time duration of critical-path. ^{0}: AND-efficient digit-serial (AEDS). ^{1}: XOR-efficient digit-serial (XEDS). ^{2}: ω-sequential multipliers with parallel output I (ω-SMPOI). ^{3}: ω-sequential multipliers with parallel output II (ω-SMPOII). ^{4}: Type I ONB structure.

$\tau^1 = T_A + (1 + \lceil \log_2 m \rceil)T_X \quad \tau^2 = 1.5Pm - 2P + 1.5m - 1 \quad \tau^3 = Pm + m + P - 1 \quad \tau^4 = Pm/2 + m + P + 1 \quad \tau^5 = 3Pm/2 + m + P - 1 \quad \tau^6 = 2T_A + (3 + \lceil \log_2(P-1) \rceil)T_X \quad \tau^7 = Pm + m + P + 1 \quad \tau^{11} = 2T_A + (3 + \lceil \log_2(P-1) \rceil)T_X \quad \tau^9 = T_A + (1 + \lceil \log_2 P + 1 \rceil)T_X \quad \tau^{10} = T_A + T_X \quad \tau^{11} = P(m+1)/d + 2(m+1) \quad \tau^{12} = T_A + (1 + \lceil \log_2 d \rceil)T_X \quad \tau^{13} = P(m+1)/d + m + 1$

$^!$: $d = 1$, where d is the number of bits of operand A fed to each PPGU during each cycle period. *: Refers to $1 < d < P$.

5.4.2 Comparison with existing digit-serial RB multipliers

The area-time complexities of proposed structures and existing structures of [13]-[14], [68], [70]-[71] for RB multiplier are listed in Table 5. For simplicity of discussion, we refer PS-I of Fig. 22 and PS-II of Fig. 24 as the case of $d = 1$, respectively.

In [13] and [14], the authors have shown that their structures outperform the previous structures in [68], [70]-[71]. Therefore we compare the performance of proposed structures only with those of [13] and [14]. PS-I and PS-II (for $d = 1$), not only involve less time complexity (shorter ACT), but also have less XOR gates than those of existing designs of [13] and [14]. PS-I and PS-II (for $1 < d < P$) require less registers, at the cost of a small increase in critical-path. And PS-III has the lowest time-complexity among all the structures listed in Table 5.

5.4.3 Comparison with existing digit-serial multipliers having a type I ONB

The complexity of RB multiplier is almost the same as that of type I ONB [68]. The area-time complexities of the proposed multipliers and architectures of [69], [74]-[77] (for which there exists a type I ONB) are shown in Table 6. Note that these complexities are estimated by substituting n for $m + 1$, according to the definition in [68].

The authors in [13] and [14] have shown that their multipliers outperform the previously proposed structures in [69], [74]-[76]. Therefore we compare our proposed structures only with [13-14] and [77]. PS-I and PS-II not only require less number of logic gates and registers (P number less XOR gates and nearly m number less registers), but also have shorter ACT compared to the structure in [77].

5.4.4 Comparison of synthesis results for FPGA implementation

We have used Altera Quartus II 12.0 and chosen Arria II GZ EP2AGZ225FF35C3 FPGA device to synthesize the proposed designs as well as the existing competing designs. The key synthesis results are obtained, in terms of area, maximum frequency and power consumption with respect to various P, Q and d. The number of adaptive look-up table (ALUT) is taken

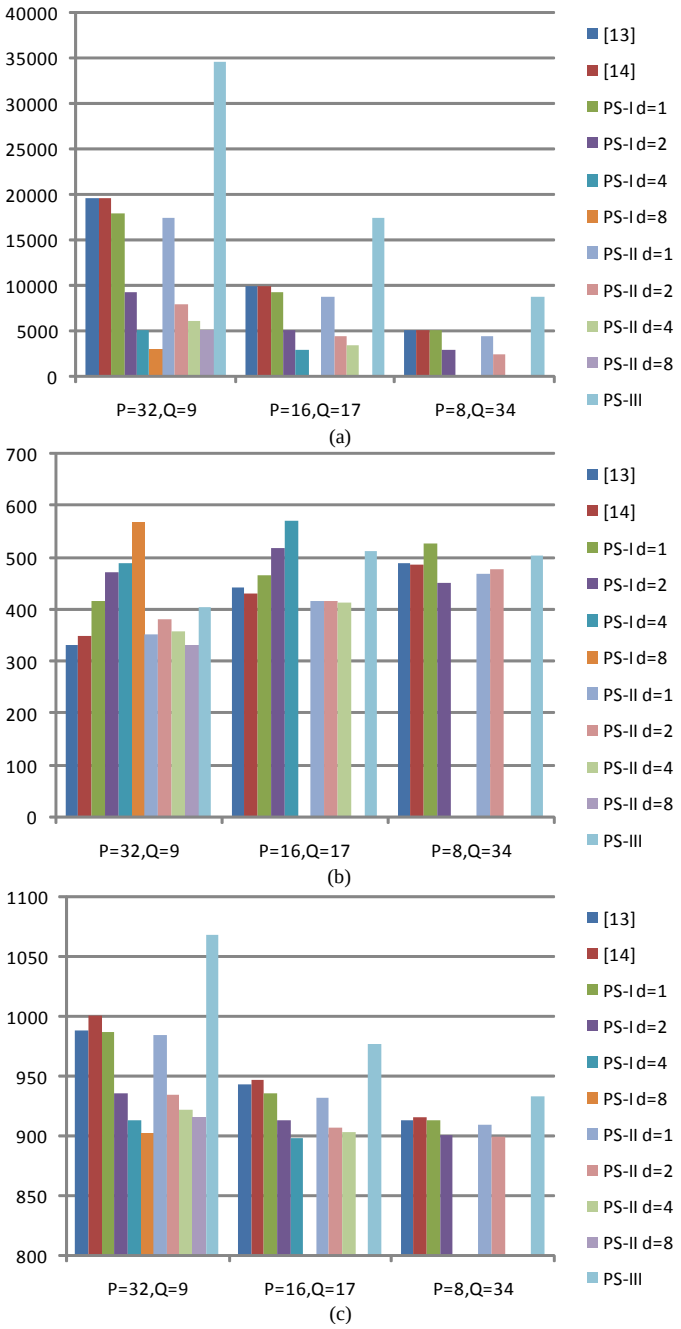

Figure 26: Comparisons of key metrics of various structures for $n = 268$. (a) Comparisons of area-complexity (number of ALUT). (b) Comparisons of maximum frequency (MHz). (c) Comparisons of power consumption (mW).

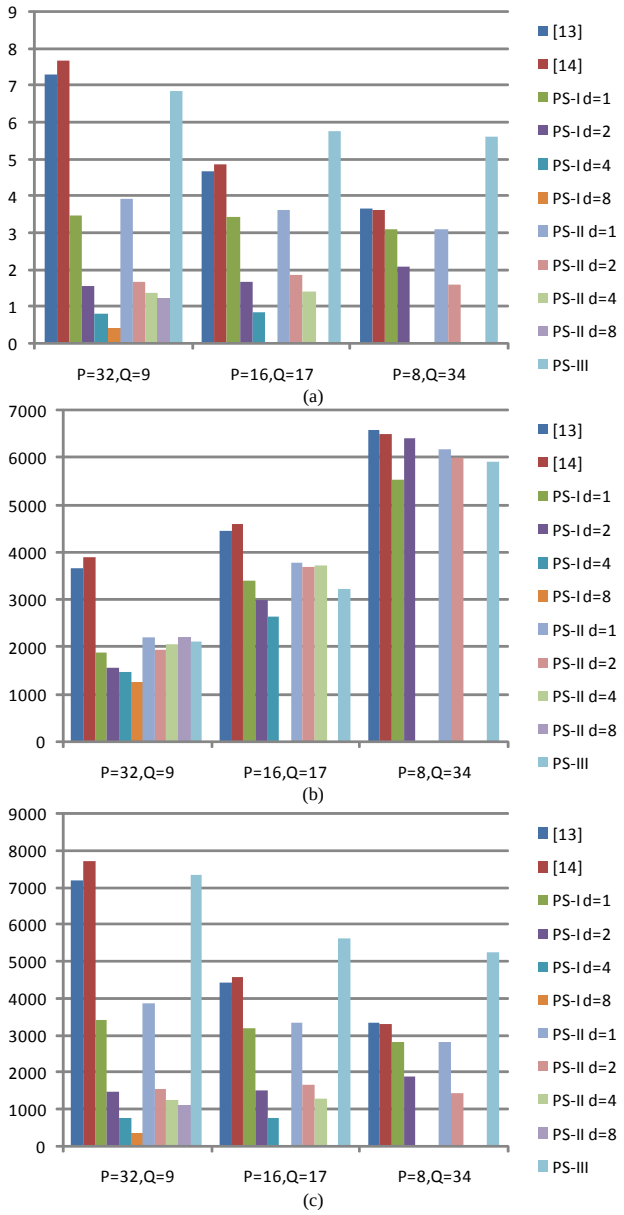

Figure 27: Comparisons of area-delay-power complexities of various structures for $n = 268$ (delay refers to the ACT of a structure. Area, delay and power are measured in number of ALUT, 10^{-4}s and mW, respectively). (a) Comparisons of area-delay product (ADP). (b) Comparisons of power-delay product (PDP). (c) Comparisons of area-delay-power product (ADPP).

as the area measure. For fair estimation, we have used the same input data and the same clock frequency (100MHz) to obtain the synthesis results using Quartus II PowerPlay Power Analyzer. The metrics, i.e., area, maximum usable clock frequency and power consumption of various structures estimated from the synthesis results are shown in Fig. 26. We have also estimated the ADP, PDP and ADPP of proposed and existing structures from the synthesis results as shown in Fig. 27, where the delay refers to the ACT estimated from the minimum data arrival time.

For a detailed comparison, we have listed the synthesis results (area, delay, power, ADP, PDP and ADPP) of proposed designs (PS-I/PS-II and PS-III) along with the best of the existing designs of [13]/[14] in Table 7, for $\{P = 32,\ Q = 9$, and $d = 8\}$, $\{P = 16,\ Q = 17$, and $d = 4\}$, and $\{P = 8,\ Q = 34$, and $d = 2\}$, respectively.

It can be seen that for FPGA implementation, the proposed structures (except PS-III) outperform the existing designs. As shown in Figs. 26, 27 and Table 7, PS-I can provide a saving of upto 94% of ADP and ADPP and 65% of PDP over the existing design of [13], for $\{P = 32,\ Q = 9$, and $d = 8\}$. Besides, as shown in Fig. 27 and Table 7, as the value of d increases, the ADP of proposed structures decreases. It is worth noting that the ALUT of Altera FPGA devices can be mapped to logic function involving multiple Boolean operations, so that the number of synthesized ALUT decreases as d increases. This feature also explains why the ADP of PS-III is worse than others.

5.4.5 Comparison of synthesis results for ASIC implementation

We have also synthesized the proposed structures and the existing structures using Synopsys Design Compiler by North Carolina State University's 45nm FreePDK [15] to obtain the area, time and power complexities of the designs. Using those synthesis results, we have plotted the area, delay and power consumption (at 1GHz) in Fig. 28, and we have calculated the ADP, PDP and ADPP of the designs (shown in Fig. 29).

As shown in Figs. 28 and 29, proposed structures outperform the existing designs. Basic structures of PS-I and PS-II ($d = 1$) have the lowest ADP and PDP among all these structures. As d increases, the ADP and PDP of proposed structures also increase a little

Table 7: Area-time-power complexities comparison of various multipliers based on FPGA implementation

Design	Area[1]	Delay[1,2]	Power[1]	ADP	PDP	ADPP
$\{P = 32,\ Q = 9$, and $d = 8\}$						
[13]	19637	3.7	989	7.3	3660	7222
PS-I	2966	1.42	902	0.42	1281	379
PS-III	34701	1.98	1069	6.87	2117	7345
$\{P = 16,\ Q = 17$, and $d = 4\}$						
[13]	9953	4.72	943	4.7	4453	4434
PS-I	2959	2.97	898	0.88	2669	791
PS-III	17485	3.31	977	5.79	3234	5658
$\{P = 8,\ Q = 34$, and $d = 2\}$						
[14]	5111	7.09	916	3.64	6496	3335
PS-II	2421	6.69	900	1.62	6021	1458
PS-III	8877	6.33	934	5.62	5910	5247

[1]: Area, delay and power are measured in number of ALUT, 10^{-4}s and mW, respectively.

[2]: Delay refers to the ACT of a structure.

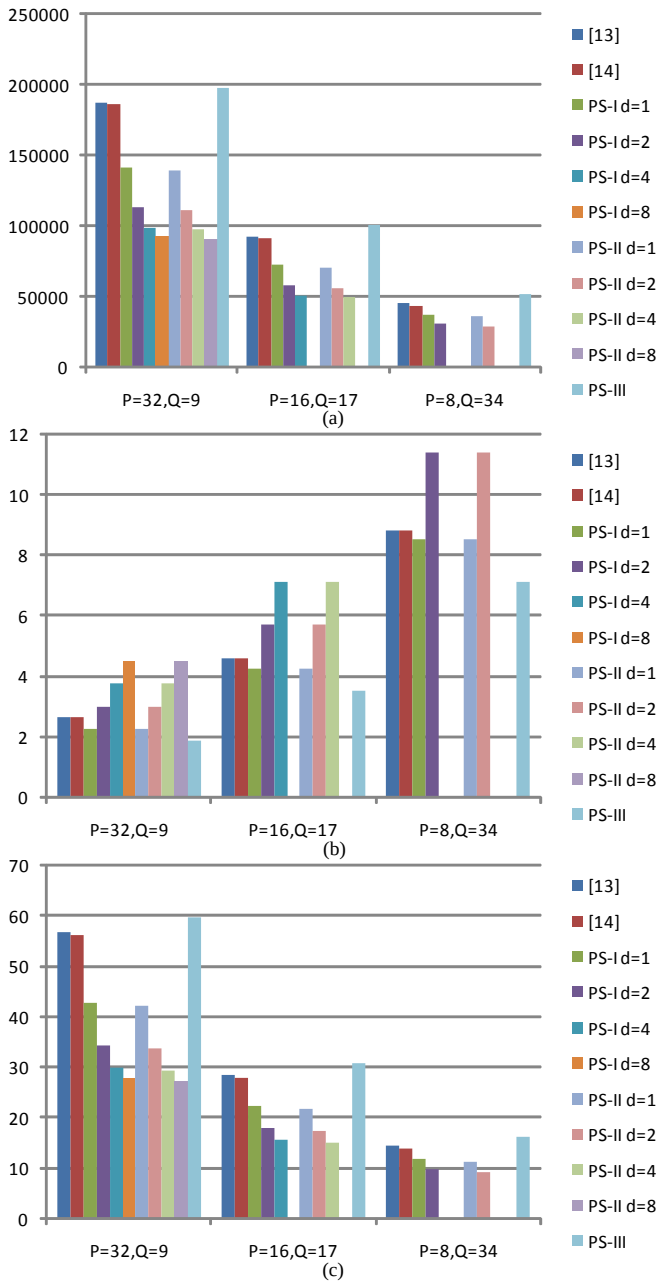

Figure 28: Comparisons of key metrics of various structures for $n = 268$. (a) Comparisons of area-complexity (μm^2). (b) Comparisons of delay (ns). (c) Comparisons of power consumption (mW).

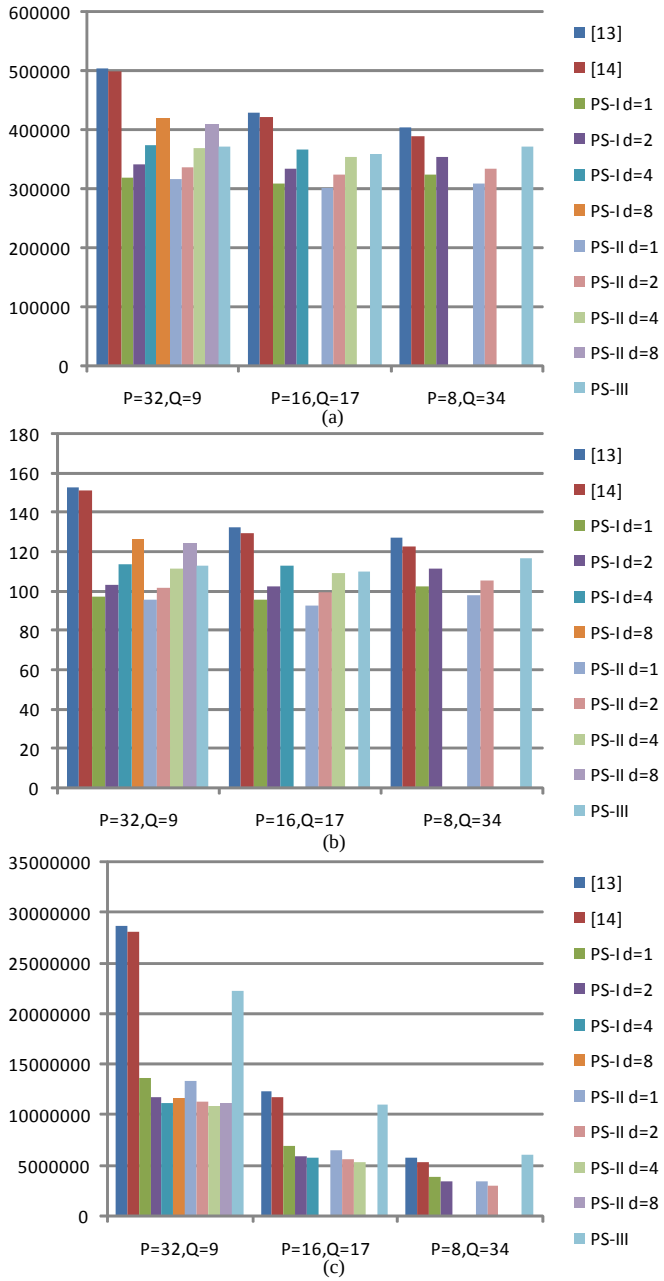

Figure 29: Comparisons of area-delay-power complexities of various structures for $n = 268$ (delay refers to the ACT of a structure. Area, delay and power are measured in μm^2, ns and mW/GHz, respectively). (a) Comparisons of area-delay product (ADP). (b) Comparisons of power-delay product (PDP). (c) Comparisons of area-delay-power product (ADPP).

while the ADPP decreases, so that the ADP and the PDP of PS-II are the lowest among all the structures for the case of $\{P = 32,\ Q = 9, \text{ and } d = 1\}$, while the ADPP of PS-II is the lowest for the case of $\{P = 32,\ Q = 9, \text{ and } d = 8\}$. For a detailed comparison, synthesis results in terms of area-delay-power complexities of proposed designs of PS-II, and PS-III; and the best of the existing designs [14] are listed in Table 8 for the case of $\{P = 32,\ Q = 9,$ and $d = 8\}$, $\{P = 16,\ Q = 17,$ and $d = 4\}$, and $\{P = 8,\ Q = 34,$ and $d = 2\}$, respectively. As shown in Figs. 28 and 29 and Table 8, especially for the case of $\{P = 32,\ Q = 9,$ and $d = 8\}$, PS-II can save at most 18% ADP, 17.8% PDP and 60% ADPP over the existing design of [14], as shown in Fig. 29 and Table 8. Moreover, as shown in Table 8, PS-III of Fig. 25 has the lowest time complexity among all the structures.

5.4.6 Design selection

From Figs. 26, 27, 28 and 29, we find that PS-I and PS-II outperform the other structures in both FPGA and ASIC platforms in terms of area, time and power complexities. Besides, because of their low area-time-power complexities and high throughput rate, PS-I and PS-II can be used in various real time applications. Especially for FPGA implementation, it is suggested to use either PS-I/II (for $1 < d < P$) based on the area constraint and speed requirement of applications. For ASIC implementation, PS-I and PS-II of Figs. 22 and 24 or PS-III of Fig. 25 are preferred for their efficiency in area-time-power complexities. For applications requiring highest throughput, PS-III of Fig. 25 is the best choice. In summary, we can choose different structures according to the requirements of different application environments.

5.5 CONCLUSION

We have proposed a novel recursive decomposition algorithm for RB multiplication to derive high-throughput digit-serial multipliers. By suitable projection of SFG of proposed algorithm and identifying suitable cut-sets for feed-forward cut-set retiming, three novel

Table 8: Area-time-power complexities comparison of various multipliers based on ASIC implementation

Design	Area[1]	Delay[1,2]	Power[1]	ADP	PDP	ADPP
$\{P = 32,\ Q = 9$, and $d = 8\}$						
[14]	185957	2.7	56.3	499852	151.3	28135
PS-II	90567	4.5	27.4	410814	124.3	11262
PS-III	197746	1.9	59.9	373742	113.1	22371
$\{P = 16,\ Q = 17$, and $d = 4\}$						
[14]	91407	4.6	28.1	422299	130.0	11884
PS-II	49676	7.1	15.3	354689	109.2	5815
PS-III	100631	3.6	31.0	359251	110.6	11130
$\{P = 8,\ Q = 34$, and $d = 2\}$						
[14]	44136	8.8	13.9	389283	122.9	5423
PS-II	29231	11.4	9.2	333932	105.4	3081
PS-III	52072	7.1	16.4	371796	117.3	6111

[1]: area, delay and power are measured in μm^2, ns and mW(1GHz), respectively.

[2]: delay refers to the ACT of a structure.

high-throughput digit-serial RB multipliers are derived to achieve significantly less area-time-power complexities than the existing ones. Moreover, efficient structures with low register-count have been derived for area-constrained implementation; and particularly for implementation in FPGA platform where registers are not abundant. The results of synthesis show that proposed structures can achieve saving of up to 94% and 60%, respectively, of ADPP for FPGA and ASIC implementation, respectively, over the best of the existing designs. The proposed structures have different area-time-power trade-off behavior. Therefore, one out of the three proposed structures can be chosen depending on the requirement of the application environments.

6.0 SINGLE AND HYBRID ARCHITECTURES FOR MULTIPLICATION OVER FINITE FIELD $GF(2^M)$ BASED ON REORDERED NORMAL BASIS

In this chapter, efficient bit- and digit-level algorithms for computing multiplication over $GF(2^m)$ based on RNB are presented. Novel high-throughput low-complexity architectures are presented based on these proposed algorithms. First of all, high-throughput bit- and digit-parallel multipliers are presented. To have an optimal balanced trade-off between area and time complexities, novel digit-serial architectures for RNB multiplication is proposed then. Finally, for the first time, a novel hybrid architecture for parallel/serial realization of finite field multiplication based on RNB is introduced. The main advantage of the novel hybrid architecture is that it offers flexible choices of throughput of parallel/serial realization of RNB multiplication while meantime it involves little hardware overhead. This feature would be a major advantage for implementing multiplication in modern/emerging reconfigurable cryptographic systems. Both theoretical comparison and practical simulation results from FPGA and ASIC realization are presented. It is shown that the proposed multipliers have significantly lower area-time-power complexity than the corresponding existing designs. Specifically, FPGA realization of the novel hybrid multiplier is detailed presented to confirm its efficiency in FPGA based reconfigurable cryptographic platforms.

6.1 INTRODUCTION

Finite field multiplication over $GF(2^m)$ is widely used in modern/emerging cryptographic systems such as ECC [1]-[3]. Due to the essential requirements of real time applications like high-throughput, low-cost and small-size, area-time-power-efficient hardware design for

field multiplication is therefore quite critical. In general, there are three bases of representation, e.g., dual basis, normal basis, and polynomial basis representations of the element of $GF(2^m)$ [4]-[9]. Meanwhile, modern cryptographic systems usually involve a number of arithmetic operations such as the multiplication, squaring and so on. It is worthy mentioning that normal basis multipliers have no hardware cost in squaring operations (only involve bits-shifting), more and more cryptographic circuits designs prefer to use the normal basis multipliers compared with the multipliers based on the other two bases [10]-[12].

There are two special classes of normal basis for which the complexity of multiplication can be minimized, namely the ONB type I and II. ONB Type II has been widely used for various cryptographic system designs. RNB is a version of ONB type II which has been proposed in [74] for efficient multiplication implementation. Later, efficient multipliers are suggested in [68] and [70] based on this basis. Very recently, two high-speed architectures for multiplication using RNB are proposed in [79].

Basically, there are three types of design styles for finite field multiplications, namely the fully-serial, fully-parallel and digit-serial. Fully-serial architecture usually suffers low-throughput and it is not frequently used. While the fully-parallel architecture has advantage like high-throughput at the cost of large area-complexity. Digit-serial architecture is widely used in many real-time cryptographic systems due to its tradeoff in area-time complexities between fully-serial and fully-parallel designs.

As technology advances, e.g., more and more gates can be fabricated in one single chip, fully-parallel realization of finite field multiplication is becoming more and more popular in many cryptographic systems due to its high-throughput capability. Therefore, high-throughput low-complexity implementation of finite field multiplier is indeed in demanding. On the other hand, emerging computing systems such as the reconfigurable cryptographic systems, require field multipliers have the ability to provide flexible input/output throughput choices to meet various real-time application requirements. While the traditional design only focusses on single style multiplier designing. Thus, designing a multiplier has multiple throughput choices with small hardware overhead would be a real challenge.

In this chapter, we present efficient bit- and digit-level algorithms for multiplication over $GF(2^m)$ based on RNB. Efficient bit- and digit-parallel (BP and DP) multipliers are

proposed first. Then, novel digit-serial (DS) multipliers are introduced to achieve optimal tradeoff between area-time complexities. Most importantly, a novel hybrid architecture for parallel/serial realization of RNB multiplication is proposed. To the best of the authors' knowledge, this is the first hybrid RNB multiplier which provides flexible choices of throughput for parallel/serial realization ever reported. The proposed hybrid multiplier also involves little hardware overhead when compared with BP multiplier. Based on its characteristics, the proposed hybrid multiplier can be used in applications such as reconfigurable cryptographic systems. To get the actual implementation results, we have used VHDL to synthesize these architectures in both FPGA and ASIC platforms for different digit sizes.

The organization of this chapter is as follows: in Section 6.2, we briefly review preliminaries of multiplication in RNB over $GF(2^m)$. In Section 6.3, efficient algorithms for bit- and digit-level realization of RNB multiplication are proposed. In Section 6.4, we present high-throughput low-complexity DS RNB multipliers. Then, we have presented a highly efficient hybrid architecture for RNB multiplication in Section 6.5, which can offer multiple throughput choices. Performance and comparison of the proposed and existing multipliers are presented in Section 6.6. And conclusion of this paper is given in Section 6.7.

6.2 PRELIMINARIES

RNB is a reordered version of an ONB type II and it is proposed firstly in [74] by Gao and Vanstone, and then it is used in efficient realization of multipliers in [78] and [70].

Define function $s[i]$ as follows, for $1 \leq i \leq m$ [10]-[11]:

$$s[i] = \begin{cases} i \bmod 2m+1 \\ (\text{if } 0 \leq i \bmod 2m+1 \leq m) \\ 2m+1-i \bmod 2m+1 \\ (\text{if } m \leq i \bmod 2m+1 \leq 2m) \end{cases} \tag{6.1}$$

Then let $A,\ B$ and the product C be $A,\ B,\ C \in GF(2^m)$. The RNB $\{x_1,\ x_2, ...,\ x_m\}$ be used to represent the field elements, so that $A,\ B,\ C$ can be presented as the following

steps (all these following equations can be written in a algorithm standard form for easy understanding and access):

$$A = \sum_{i=1}^{m} a_i x_i,\ B = \sum_{i=1}^{m} b_i x_i,\ C = \sum_{i=1}^{m} c_i x_i, \tag{6.2}$$

where $a_i,\ b_i,\ c_i \in \{0,\ 1\},\ \text{for } 1 \leq i \leq m.$

Then, we can have [11]

$$c_i = \sum_{j=1}^{m} a_j b_{s[i+j]} + \sum_{j=1}^{m} a_j b_{s[j-i]} \tag{6.3}$$

for $i,\ j \in \{1,\ 2,\ ...,\ m\}$, and b_0 is defined as 0.

6.3 PROPOSED BIT- AND DIGIT-PARALLEL RNB MULTIPLIERS

6.3.1 Proposed bit-parallel (BP) RNB multiplication algorithm

First of all, we can rewrite (6.3) into another matrix-vector form as shown in the following steps

$$\begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} b_{s[2]} & \cdots & b_{s[m+1]} \\ \vdots & \ddots & \vdots \\ b_{s[m+1]} & \cdots & b_{s[2m]} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} b_{s[0]} & \cdots & b_{s[m-1]} \\ \vdots & \ddots & \vdots \\ b_{s[1-m]} & \cdots & b_{s[0]} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \tag{6.4}$$

Then, according to (6.1), (6.4) can be expressed as another matrix-vector multiplication form:

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} b_2 & b_3 & \cdots & b_m \\ b_3 & b_4 & \cdots & b_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_m & b_{m-1} & \cdots & b_1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & \cdots & b_{m-1} \\ b_1 & b_0 & \cdots & b_{m-2} \\ b_2 & b_1 & \cdots & b_{m-3} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m-1} & b_{m-2} & \cdots & b_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{6.5}$$

Moreover, we can also express second matrix of (6.5) into the form as

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} b_2 & b_3 & \cdots & b_m \\ b_3 & b_4 & \cdots & b_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_m & b_{m-1} & \cdots & b_1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} b_{m-1} & b_{m-2} & \cdots & b_0 \\ b_{m-2} & b_{m-3} & \cdots & b_1 \\ \vdots & \vdots & \ddots & \vdots \\ b_0 & b_1 & \cdots & b_{m-1} \end{bmatrix} \begin{bmatrix} a_m \\ a_{m-1} \\ \vdots \\ a_1 \end{bmatrix} \tag{6.6}$$

The equation (6.6) can be seen as the addition of two matrix-vector multiplications, and can be rewritten as one matrix-vector multiplication if we add some additional bits.

Here we define $\{c_{m+1},\ c_{m+2},\ \ldots,\ c_{2m},\ c_{2m+1}\}$ and $\{a_{m+1}\}$ as additional bits, such that we can have the equation in Fig. 30, where the bits in the dashed boxes are extra bits

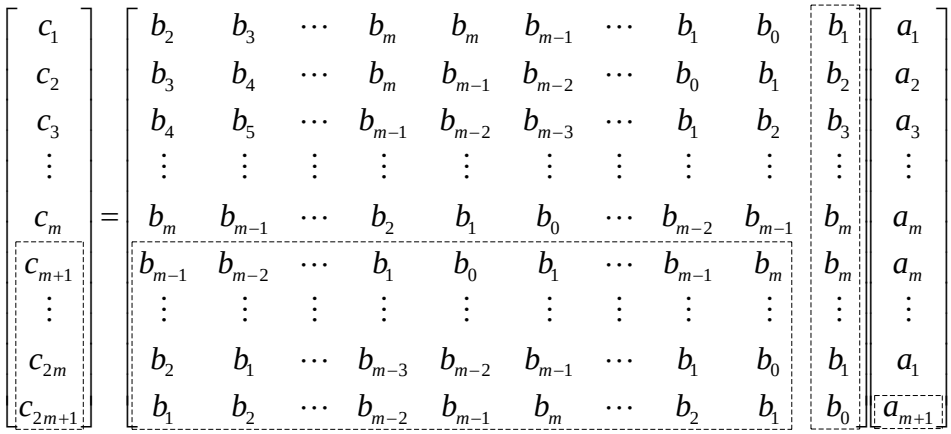

Figure 30: Extended expression of equation (6.6), where the bits in the dashed box are extra added bits.

added to meet the multiplication requirement (these bits are filled to meet multiplication requirement, but they won't be really used in real structure implementation), where

$$
\begin{aligned}
& a_{m+1} = 0 \\
& c_{m+1} = b_{m-1}a_1 + \cdots + b_1 a_m + \cdots + b_m a_{m+1} \\
& \cdots \quad \cdots \quad \cdots \\
& c_{2m} = b_2 a_1 + \cdots + b_{m-1} a_m + \cdots + b_0 a_{m+1} \\
& c_{2m+1} = b_1 a_1 + \cdots + b_m a_m + \cdots + b_1 a_{m+1}
\end{aligned}
\tag{6.7}
$$

Define $\{x_{m+1},\ x_{m+2},\ \ldots,\ x_{2m+1}\}$ as extended RNB, and then according to the equation in Fig. 30 we can also define as followings

$$
\begin{aligned}
\overline{C} &= \sum_{i=1}^{2m+1} c_i x^i = c_1 x_1 + \cdots c_{m+1} x_{m+1} + \cdots c_{2m+1} x_{2m+1} \\
\overline{B} &= \sum_{i=1}^{2m+1} \overline{b}_i x^i = b_1 x_1 + \cdots b_m x_{m+1} + \cdots b_0 x_{2m+1} \\
\overline{A} &= \sum_{i=1}^{2m+1} \overline{a}_i x^i = a_1 x_1 + \cdots a_m x_{m+1} + \cdots a_{m+1} x_{2m+1}
\end{aligned}
\tag{6.8}
$$

From (6.8), based on the expression of $\overline{B}$ and Fig. 30, we can also define as following equation steps:

$$\begin{aligned}
&\overline{B}^1 = \sum_{i=1}^{2m+1} \overline{b}_i^1 x^i \\
&= b_2 x_1 + \cdots + b_{m-1} x_{m+1} + \cdots + b_1 x_{2m+1} \\
&\overline{B}^2 = \sum_{i=1}^{2m+1} \overline{b}_i^2 x^i \\
&= b_3 x_1 + \cdots + b_{m-2} x_{m+1} + \cdots + b_2 x_{2m+1} \\
&\quad \cdots \quad \cdots \quad \cdots \\
&\overline{B}^{2m} = \sum_{i=1}^{2m+1} \overline{b}_i^{2m} x^i \\
&= b_0 x_1 + \cdots + b_m x_{m+1} + \cdots + b_1 x_{2m+1}
\end{aligned} \tag{6.9}$$

where

$$\begin{aligned}
&\overline{b}_{2m}^{i+1} = \overline{b}_1^i \\
&\overline{b}_j^{i+1} = \overline{b}_{j+1}^i, \text{ for } 1 \le j \le 2m-1
\end{aligned} \tag{6.10}$$

The recursions on (6.10) can be extended further to have

$$\overline{b}_j^{i+v} = \begin{cases} \overline{b}_{2m-v+j}^i, & \text{for } 1 \le j \le v \\ \overline{b}_{j+v}^i, & \text{otherwise} \end{cases} \tag{6.11}$$

where $1 \le v \le 2m-1$, such that all $2m+1$ bits of $\overline{B}^1$, $\overline{B}^2$, ..., $\overline{B}^{2m}$ are the same as $\overline{B}$ and can be obtained through bit-shifting operation from $\overline{B}$, respectively.

And from Fig. 30 we can have

$$\overline{C} = \sum_{i=1}^{2m} \overline{B}^i \overline{a}_i = \overline{B}^1 a_1 + \overline{B}^2 a_2 + \cdots + \overline{B}^{2m} a_1 \tag{6.12}$$

for $a_{m+1} = 0$ as defined above, and this definition can be used in rest sections including subsections of this chapter.

Based on the above equations, similarly, we also can define as the following equation steps:

$$\begin{aligned}
\overline{C}_S &= \sum_{i=1}^{m} c_i x_i = c_1 x_1 + \cdots + c_m x_m = C \\
\overline{B}_S^1 &= \sum_{i=1}^{m} \overline{b}_{Si}^1 x_i = b_2 x_1 + \cdots + b_m x_m \\
\overline{B}_S^2 &= \sum_{i=1}^{m} \overline{b}_{Si}^2 x_i = b_3 x_1 + \cdots + b_{m-1} x_m \\
&\cdots \quad \cdots \quad \cdots \\
\overline{B}_S^{2m} &= \sum_{i=1}^{m} \overline{b}_{Si}^{2m} x_i = b_0 x_1 + \cdots + b_{m-1} x_m \\
\overline{A}_S &= \sum_{i=1}^{m} a_i x_i = a_1 x_1 + \cdots + a_m x_m = A
\end{aligned} \tag{6.13}$$

where

$$\begin{aligned}
&\overline{b}_{Si}^1 = \overline{b}_i^1, \quad \text{for } 1 \le i \le m \\
&\overline{b}_{Si}^2 = \overline{b}_i^2, \quad \text{for } 1 \le i \le m \\
&\cdots \quad \cdots \quad \cdots \\
&\overline{b}_{Si}^{2m} = \overline{b}_i^{2m}, \quad \text{for } 1 \le i \le m \\
&a_i = \overline{a}_i, \quad \text{for } 1 \le i \le m
\end{aligned} \tag{6.14}$$

that $\overline{B}_S^1$, $\overline{B}_S^2$, $\ldots$, $\overline{B}_S^{2m}$ can be obtained through bit-selecting operation from $\overline{B}^1$, $\overline{B}^2$, $\ldots$, $\overline{B}^{2m}$, respectively. According to Fig. 30, the former RNB multiplication can be rewritten as

$$\begin{aligned}
C = AB &= \overline{C}_S \\
&= \sum_{i=1}^{m} \overline{B}_S^i a_i + \sum_{i=m+1}^{2m} \overline{B}_S^i a_{2m+1-i} \\
&= \sum_{i=1}^{2m} \overline{B}_S^i \overline{a}_i
\end{aligned} \tag{6.15}$$

The proposed BP multiplication scheme based on (6.15) is described in Algorithm 6.1.

Algorithm 6.1 Proposed BP multiplication algorithm

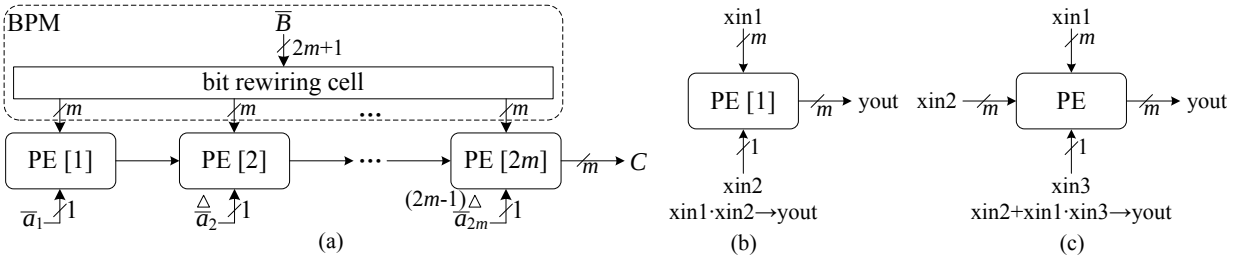

Figure 31: Proposed bit-parallel (BP) RNB multiplication architecture, where $\triangle$ stands for unit delay. (a) Proposed architecture. (b) Function of the first PE. (c) Function of the regular PE (PE[2] to PE[$2m$]).

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$, $\overline{A}$

2. Multiplication step

   2.1. obtain $\overline{B}^1_S$, $\overline{B}^2_S$, …, $\overline{B}^{2m}_S$, from $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$ through bit-selecting operation

   2.2. for $i = 1$ to $2m$

   2.3. $C = \overline{C}_S = \sum_{i=1}^{2m} \overline{B}^i_S \overline{a}_i$

   2.4. end for

We can also rewrite (6.15) into another form as

$$\begin{aligned} C = AB &= \overline{C}_S \\ &= \sum_{i=1}^{m} \overline{B}^i_S a_i + \sum_{i=m+1}^{2m} \overline{B}^i_S a_{2m+1-i} \\ &= \sum_{i=1}^{m} (\overline{B}^i_S a_i + \overline{B}^{i+m}_S a_{m+1-i}) \\ &= \sum_{i=1}^{m} (\overline{B}^i_S + \overline{B}^{2m-i+1}_S) a_i \end{aligned} \tag{6.16}$$

Based on (6.16), we can have another BP algorithm as described in Modified Algorithm 6.1.

Modified Algorithm 6.1 Proposed BP multiplication algorithm

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, ..., $\overline{B}^{2m}$, $\overline{A}$

2. Multiplication step

   2.1. obtain $\overline{B}_S^1$, $\overline{B}_S^2$, $\ldots$, $\overline{B}_S^{2m}$, from $\overline{B}^1$, $\overline{B}^2$, ..., $\overline{B}^{2m}$ through bit-selecting operation

   2.2. for $i = 1$ to m

   2.3. $C = \overline{C}_S = \sum_{i=1}^{m} (\overline{B}_S^i + \overline{B}_S^{2m-i+1}) a_i$

   2.4. end for

6.3.2 Proposed BP architecture for RNB multiplier

The proposed architecture for BP multiplier based on Algorithm 6.1 is shown in Fig. 31. It consists of $2m$ processing elements (PE)s along with a bit-permutation module (BPM). The BPM performs rewiring of bits (executed by bit rewiring cell) of operand $\overline{B}$ to feed its output to $2m$ PEs according to Algorithm 6.1. Function of the first PE (PE[1]) is described in Fig. 31(b), while the function of regular PEs (PE[2] to PE[$2m$]) is depicted in Fig. 31(c). The bits of operand $\overline{A}$ are delayedfed to the individual PEs respectively to meet the data-path requirement. During each cycle, a regular PE performs an AND operation of a bit of operand $\overline{A}$ with the input polynomial $\overline{B}_S^i$ followed by bit-by-bit XOR of the result of AND operations with the complement of the accumulated result available to the PE from its left.

According to Modified Algorithm 6.1, we can have two kinds of modified architectures (the number of PEs in the structures is reduced), namely MBP-I and MBP-II, as shown

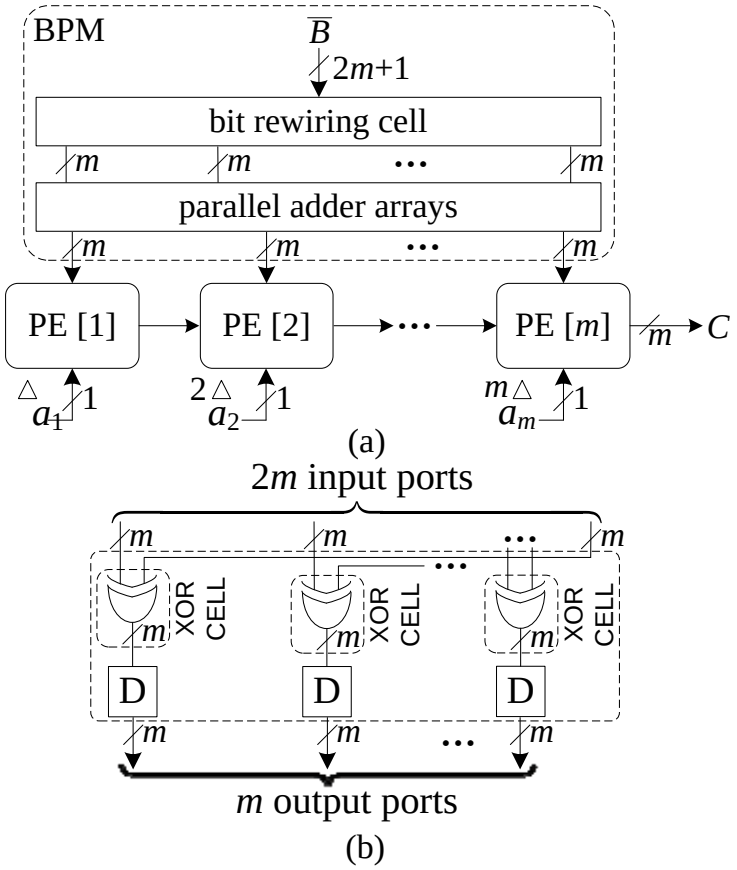

Figure 32: Proposed modified bit-parallel-I (MBP-I) RNB multiplication architecture, where $\triangle$ stands for unit delay. (a) Modified architecture. (b) Internal structure of parallel adder arrays.

later. First of all, we can have the modified bit-parallel-I (MBP-I) architecture based on Modified Algorithm 6.1 as shown in Fig. 32, where the number of PEs is reduced from $2m$ to m at the cost of extra parallel adder arrays in the BPM. To meet the data processing requirement, bits of operand A fed to each PE have one more delay than that of Fig. 31. The detail design of parallel adder arrays is shown in Fig. 32(b), which consists of m XOR cells and equal number of registers. Compare with Fig. 31, MBP-I structure of Fig. 32 has less area complexity since m PEs are reduced. Thus, we choose MBP-I of Fig. 32 as one of our preferred designs.

The detailed designs of proposed MBP-II multiplier shown in Fig. 33, where the registers in parallel adder arrays are removed to reduce the area-complexity further at the cost of minor

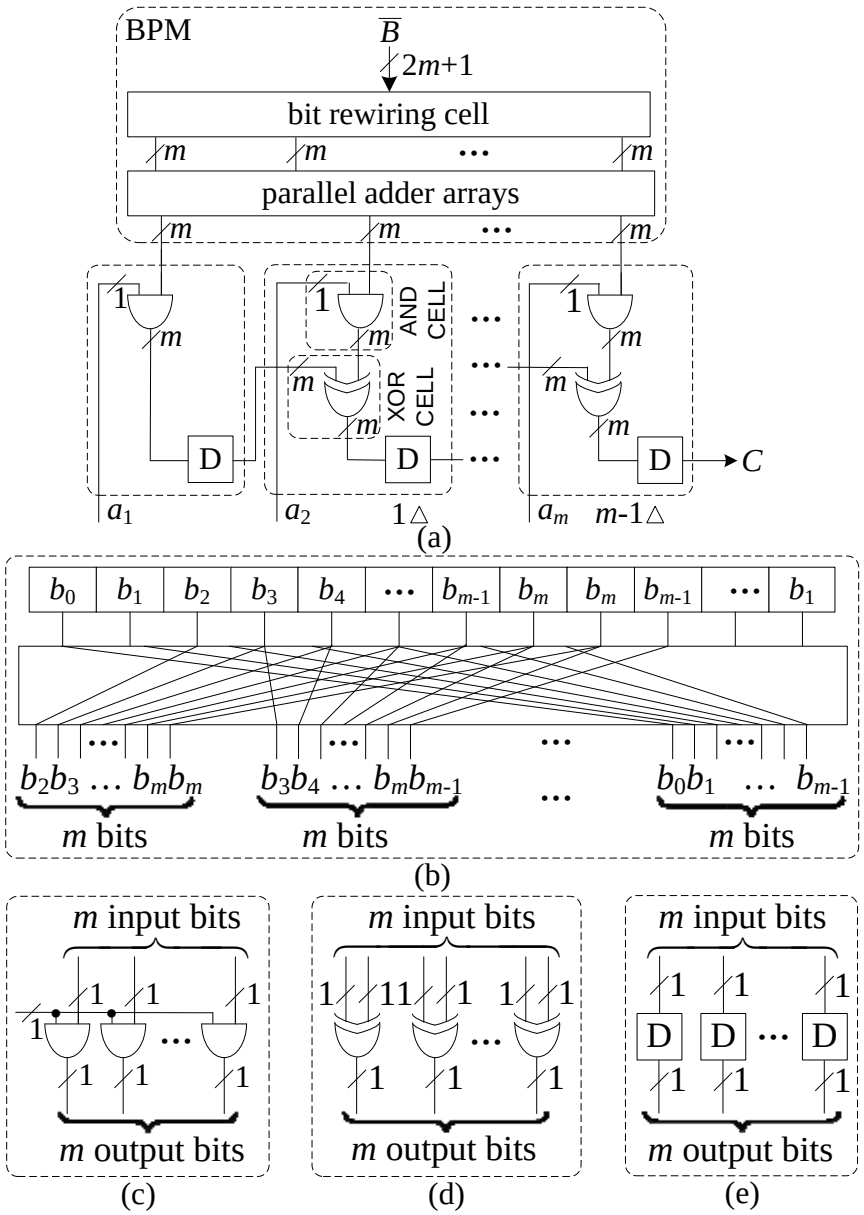

Figure 33: Detailed design of proposed MBP-II multiplier, where "D" denotes a register cell and $\triangle$ stands for unit delay. (a) Design of proposed architecture. (b) Design of BPM. (c) Structure of the AND cell. (d) Structure of the XOR cell. (e) Structure of the register cell.

increase in critical-path. $(2m+1)$ bits of operand $\overline{B}$ originally extended from B are preloaded in $(2m+1)$ bit-registers, and after that yields corresponding $2m$ outputs to parallel adder arrays to produce m outputs to PEs according to Modified Algorithm 6.1. A regular PE,

as shown in Fig. 33(a), consists of three cells: an AND cell, an XOR cell and a register cell. Structure and function of AND cell, XOR cell and register cell are shown in Fig. 33(c), (d) and (e), respectively. AND cell consists of m AND gates working in parallel to perform the bit-multiplication operations of the input a_i with the corresponding bit-shifted forms of operand $\overline{B}_S^i + \overline{B}_S^{2m-i+1}$, for $1 \leq i \leq 2m$. XOR cell consists of m XOR gates to perform bit-by-bit XOR operations. The output of the XOR cell is then latched out to the next PE in the next clock cycle. The duration of minimum cycle period of the structure in Fig. 33 is $(T_A + 2T_X)$. The proposed bit-parallel design gives the first output of desired product m cycles after the pair of operands are fed to the structure, while the successive outputs are produced in each cycle thereafter.

In real-time applications, MBP-I and -II can be chosen according to specific requirement, since MBP-I has advantage on critical-path, while MBP-II has merit on area-complexity. For simplicity of discussion, in the following sections, we all refer them as MBP, except in the comparison section.

6.3.3 Proposed digit-parallel (DP) architecture for RNB multiplier

Using the architectures of Fig. 33, we derive here the proposed digit-level parallel architecture for implementation of RNB multiplier over $GF(2^m)$, where bits of operand A are fed digit-by-digit to the corresponding PEs of the architecture. Define d as the parallel digit-size (number of bits of operand A fed to each PE during each cycle period), to obtain the structure for $d = 2$, a pair of PEs of the architecture of Fig. 33(a) can be merged to form a new PE as shown in Fig. 34(a) and (b). Likewise, the internal structure of merged PE (shown in Fig. 34(d)) is the merged form of two regular PE of bit-parallel architecture (shown in Fig. 34(c)), which consists of two AND cells, two XOR cells (the first PE requires only one XOR cell) and one register cell. The functions of AND cell, XOR cell and register cell are the same as those described in Fig. 34. The design detail of proposed digit-parallel RNB multiplier consisting of m PEs is shown in Fig. 34(e). Two bits of operand A are fed to the AND cell in parallel and the results are fed as inputs first to two parallel XOR cell. The partial result available from left is then added together with the outputs of two AND cells of PE

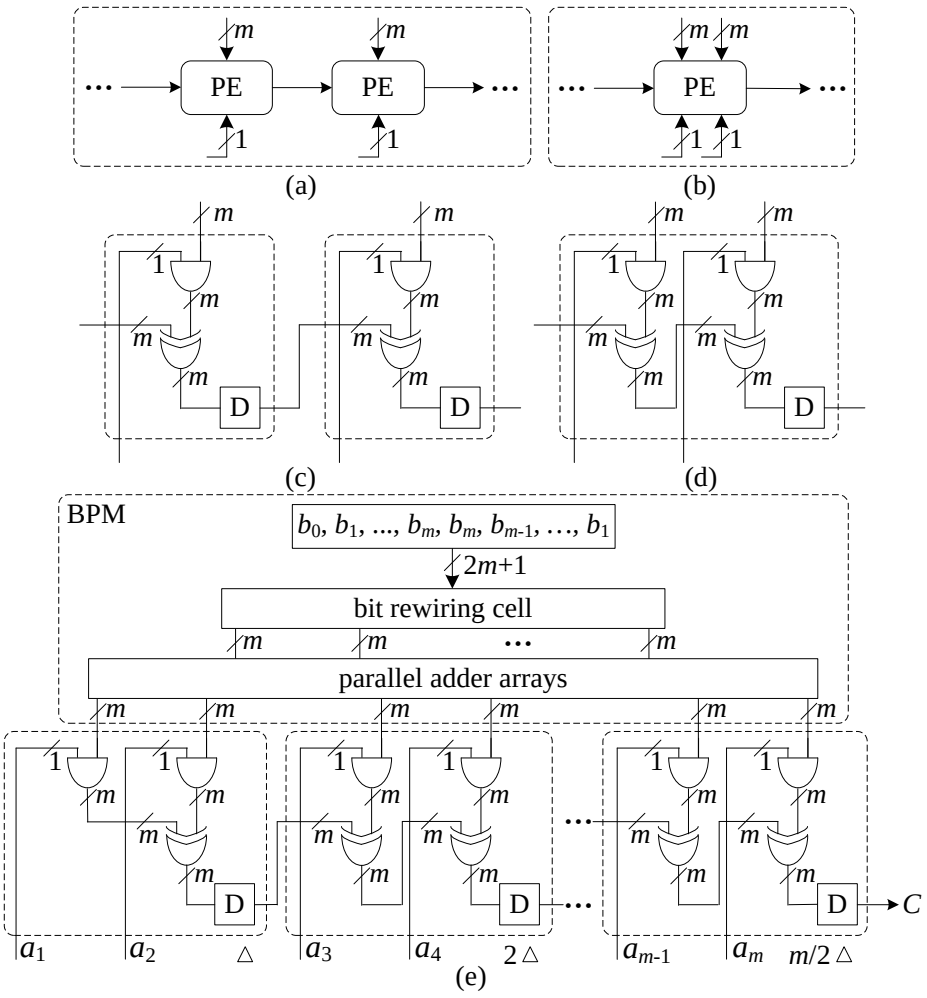

Figure 34: Proposed digit-parallel (DP) architecture for $d = 2$. (a) Two regular PEs of Fig. 31. (b) Merged PE. (c) Internal structure of two regular PEs of Fig. 31. (d) Internal structure of merged PE. (e) Design detail of proposed RNB multiplier for $d = 2$.

to generate the output to be latched out to next PE. The critical-path of the structure of Fig. 34(e) amounts to $(T_A + 2T_X)$. The first output of desired product is available from this architecture after a latency of m cycles, while the successive outputs are available thereafter in each cycle of duration $(T_A + 2T_X)$. The technique used to derive the architecture for $d = 2$ may be extended for any value of d, to obtain an architecture consisting of (m/d) PEs.

The proposed DP realization strategy, on one hand, can result in reduction of the number of registers of the multiplier; on the other hand, can significantly enhance the efficiency of

FPGA implementation: the strategy is quite useful for FPGA-based implementation since value of d can be chosen appropriately, such that the logic gates contained in a PE could be mapped to a basic unit of FPGA (on increase on critical path) to achieve low area-time-complexities implementation (number of registers is reduced).

6.3.4 Proposed low-latency bit- and digit-parallel (LBP) (LDP) architectures for RNB multiplier

MBP architectures presented in Figs. 32 and 33 provides very high throughput and involves low area-time complexity. However, it has latency of m cycles which could be too high for real-time applications, particularly when the order of the finite field m is large. While in the DP architecture of Fig. 34, the latency in terms of the number of cycles is less, but the critical-path is relatively high. Keeping these in view, we derive here reduced latency high-throughput implementations of RNB multiplier.

For any value of m where exists the ONB type II, let Q and P be two integers such that $m = QP + r$, where $0 \leq r < P$. For simplicity of discussion, we can assume $r = 0$ (and can append $(Q - r)$ number of zeros to the operands to satisfy $m = QP$, when $r \neq 0$) to have the following steps. We can first decompose extended input operand A into Q number of bit-vectors A_l for $l = 0, 1, \ldots, Q - 1$, as follows:

$$\begin{aligned} A_0 &= [a_1 \ a_{Q+1} \cdots \ a_{m-Q+1}] \\ A_1 &= [a_2 \ a_{Q+2} \cdots a_{m-Q+2}] \\ &\cdots \ \cdots \ \cdots \\ A_{Q-1} &= [a_Q \ a_{2Q} \ \cdots \ a_m] \end{aligned} \tag{6.17}$$

Define $(\overline{B}_S^i + \overline{B}_S^{2m-i+1})$ of (6.16) as $\overline{B}_{S\{M\}}^i$. Then similarly, we can generate Q number of shifted operand vectors B_l for $l = 0, 1, \cdots, Q - 1$, as follows:

$$\begin{aligned} B_0 &= [\overline{B}_{S\{M\}}^1 \ \overline{B}_{S\{M\}}^{Q+1} \cdots \ \overline{B}_{S\{M\}}^{m-Q+1}] \\ B_1 &= [\overline{B}_{S\{M\}}^2 \ \overline{B}_{S\{M\}}^{Q+2} \cdots \ \overline{B}_{S\{M\}}^{m-Q+2}] \\ &\cdots \ \cdots \ \cdots \\ B_{Q-1} &= [\overline{B}_{S\{M\}}^Q \ \overline{B}_{S\{M\}}^{2Q} \cdots \ \overline{B}_{S\{M\}}^m] \end{aligned} \tag{6.18}$$

Based on the above discussions, the product $C = AB$ can thus be rewritten as the following steps:

$$\begin{aligned} & C = AB \\ & = B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T \end{aligned} \tag{6.19}$$

where

$$\begin{aligned} & B_0 A_0^T = \\ & \overline{B}^1_{S\{M\}} a_1 + \overline{B}^{Q+1}_{S\{M\}} a_{Q+1} \cdots + \overline{B}^{m-Q+1}_{S\{M\}} a_{m-Q+1} \\ & B_1 A_1^T = \\ & \overline{B}^2_{S\{M\}} a_2 + \overline{B}^{Q+2}_{S\{M\}} a_{Q+2} \cdots + \overline{B}^{m-Q+2}_{S\{M\}} a_{m-Q+2} \\ & \cdots \quad \cdots \quad \cdots \\ & B_{Q-1} A_{Q-1}^T = \\ & \overline{B}^Q_{S\{M\}} a_Q + \overline{B}^{2Q}_{S\{M\}} a_{2Q} \cdots + \overline{B}^m_{S\{M\}} a_m \end{aligned} \tag{6.20}$$

Then, we can have proposed low-latency bit-parallel multiplication scheme based on (6.19) as described in Algorithm 6.2.

Algorithm 6.2 Proposed low-latency bit-parallel multiplication algorithm

---

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$, $\overline{A}$

2. Multiplication step

   2.1. obtain $\overline{B}^1_S$, $\overline{B}^2_S$, …, $\overline{B}^{2m+1}_S$, from $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$ through bit-selecting operation

   2.2. for $i = 1$ to m

   2.3. $C = B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T$

   2.4. end for

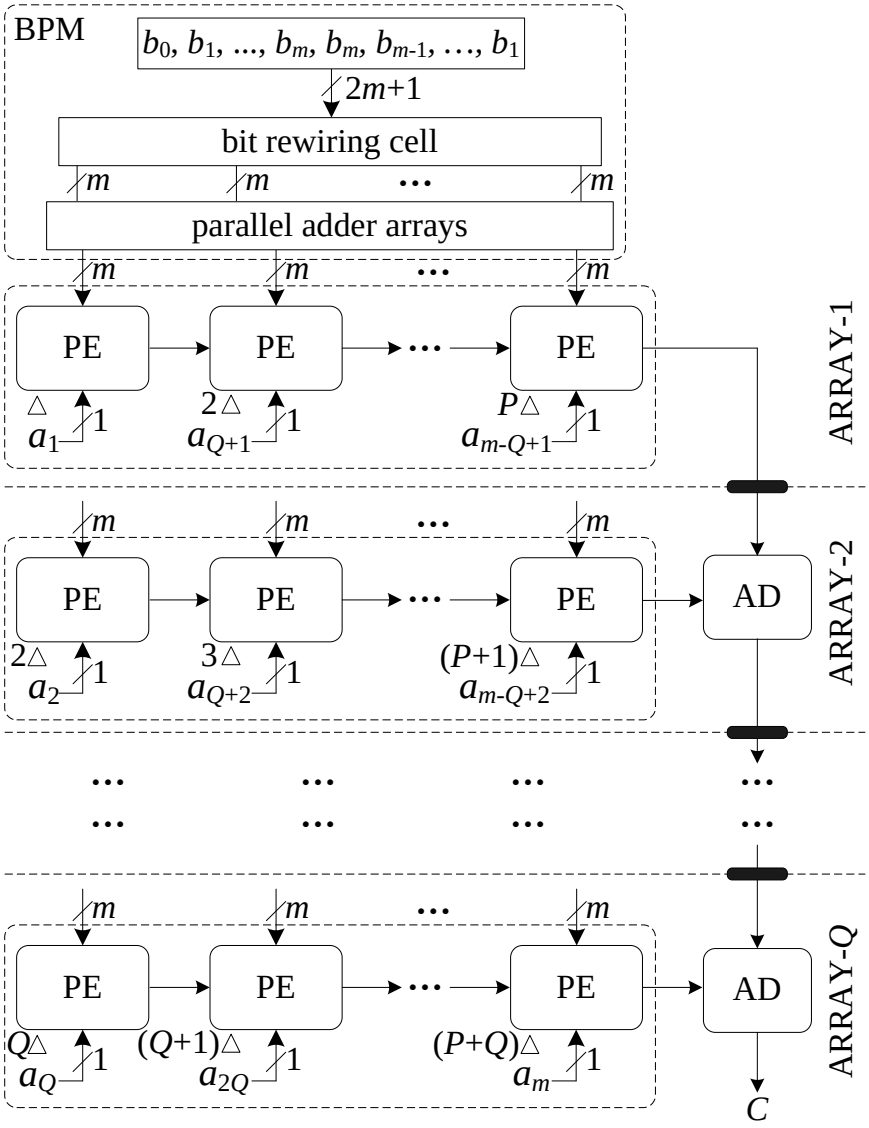

Figure 35: Proposed low-latency bit-parallel (LBP) architecture for RNB multiplier.

The proposed low-latency bit-parallel (LBP) structure for RNB multiplier based on Algorithm 6.2 is shown in Fig. 35. It consists of Q arrays, where each of the arrays consists of P PEs, and an addition cell (AD) (the first array does not require any AD). Function of the PEs of this structure is the same as that depicted in Fig. 33, while the AD performs finite field addition of its input available from the top with the input available from the left and latches out the output to its adjacent AD downward (each AD consists of m XOR gates to

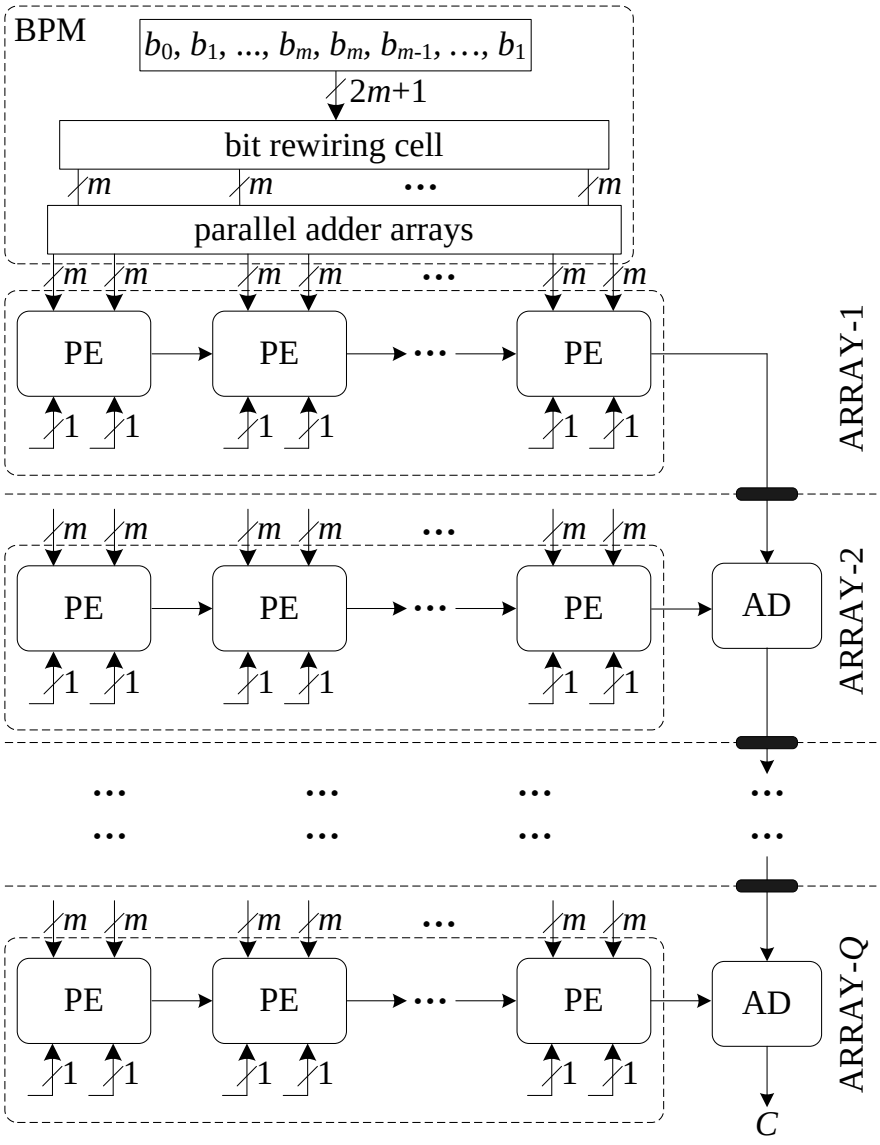

Figure 36: Proposed LBP architecture for $d = 2$.

perform the bit-by-bit operations of the two m-bit input words). The input bits of operands A fed to PE in each array are in staggered manner to meet the timing requirement in systolic pipeline. After $(P + 1)$ cycles, array-1 generates the first partial result $B_0 A_0^T$ and feeds that to the AD of array-2 to be added with the partial result $B_1 A_1^T$ (array-1 has one cycle delay). Each of the successive ADs then generates the partial result in subsequent cycles to produce the output C after a latency of $(P + Q)$ cycles at the last AD of array-Q. After the latency

period, it gives one output word in every successive cycle. And the duration of cycle period is the same as that of MBP.

Using the strategy of Fig. 34, the structure of Fig. 35 can be modified as shown in Fig. 36 for $d = 2$, where the critical path and throughput are the same as those of Fig. 34. Similarly, it could be easily extended to higher values of d to have similar structures.

6.4 PROPOSED DIGIT-SERIAL (DS) RNB MULTIPLIER

In this section, we first give a brief review of existing digit-serial (DS) RNB multiplication algorithm, and then we present our proposed digit-serial algorithm as well as the structure.

6.4.1 Brief review of existing DS RNB multiplier

In the recently proposed RNB multipliers of [79], both operands A and B are decomposed into a number of blocks to achieve DS multiplication, and after that the partial products corresponding to these blocks are added together to obtain the desired product word. The existing DS RNB multiplication algorithm is stated as follows:

Existing Algorithm Existing DS multiplication algorithm [79]

Inputs: $A = (a_1, \ldots, a_m)$, $B = (b_1, \ldots, b_m)$ are two RNB representation elements, and $t = \lceil m/w \rceil$

Output: $C = A \cdot B = (c_0, \ldots, c_m)$

1. Initialization: $e_{i,k}^{(0)} = 0, g_{i,k}^{(0)} = 0$, for $i = 1, \ldots, m$ and $k = 0, \ldots, t-1$

2. Compute in parallel for all $i = 1, 2, \ldots, m$

3. Compute in parallel for all $k = 0, 1, 2, \ldots, t-1$

4. Compute in serial for $f = 1, 2, \ldots, w$

(Steps 5, 6 and Steps 7, 8 are computed in parallel)

5. $tem1 := a_{kw+f} b_{s(i+kw+l)}$

6. $e_{i,k}^{(f)} := e_{i,k}^{(f-1)} + tem1$

7. $tem2 := a_{kw+f} b_{s(i-kw-l)}$

8. $g_{i,k}^{(f)} := g_{i,k}^{(f-1)} + tem2$

9. End

(The summation of $2t$ terms in Step 10 is performed in parallel)

10. $c_i = \sum_{k=0}^{\lceil m/w \rceil - 1} [e_{i,k}^{(w)} + g_{i,k}^{(w)}]$

11. End

12. End

Steps 5, 6, 7 and 8 of existing algorithm refer to the computation of digit-wise partial products for the DS multiplication where operands A and B are decomposed into a number of digits, and step 10 refers to the addition of those partial products to compute the product word.

Although the existing algorithm of [79] is the most efficient one out of all reported algorithms for DS multiplication, we find that the hardware utilization efficiency and throughput of existing structures of [70] could be improved further by efficient design of algorithm and architecture. Particularly, due to its larger number of digit-wise partial products (steps 5, 6, 7 and 8), and extra time for addition of those partial products (step 10), which not only increases the average computation time (ACT) to perform the multiplication but also involves extra hardware resources for storage and addition of larger number of partial products.

6.4.2 Proposed DS RNB multiplier

To derive efficient DS structures with efficient tradeoff between area and time complexities, first of all, let us rewrite (6.19) as Q inner-products of vectors A_l and B_l for $l = 0, 1, \cdots, Q-1$:

$$
\begin{aligned}
C &= AB \\
&= B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T \\
&= \sum_{l=0}^{Q-1} B_l A_l^T = \sum_{l=0}^{Q-1} \overline{C_l}
\end{aligned}
\tag{6.21}
$$

where $\overline{C_l}$ denotes

$$
\overline{C_l} = B_l A_l^T \tag{6.22}
$$

Note that each A_l for $l = 0, 1, \cdots, Q-1$ is a P-point bit-vector and each B_l for $l = 0, 1, \cdots, Q-1$ is a P-term operand-vector. From (6.21) and (6.22) we can find that the desired multiplication can be performed by Q cycles of successive accumulation of $\overline{C_l}$ for $l = 0, 1, \cdots, Q-1$, while each $\overline{C_l}$ can be computed as $\overline{C_l} = \sum_{h=0}^{P-1} \overline{B}_{S\{M\}}^{l+hQ} a_{l+hQ}$. The proposed digit-serial multiplication algorithm based on (6.21) and (6.22) is described in Algorithm 6.3.

Algorithm 6.3 Proposed DS multiplication algorithm

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$, $\overline{A}$

   1.2 $D = 0$

2. Multiplication step

   2.1. obtain $\overline{B}_S^1$, $\overline{B}_S^2$, …, $\overline{B}_S^{2m+1}$, from $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$ through bit-selecting operation

   2.2. for $l = 0$ to $Q-1$

   2.3. for $h = 0$ to $P-1$

   2.4. $D = D + B_l A_l^T$

   2.5. end for

   2.6. end for

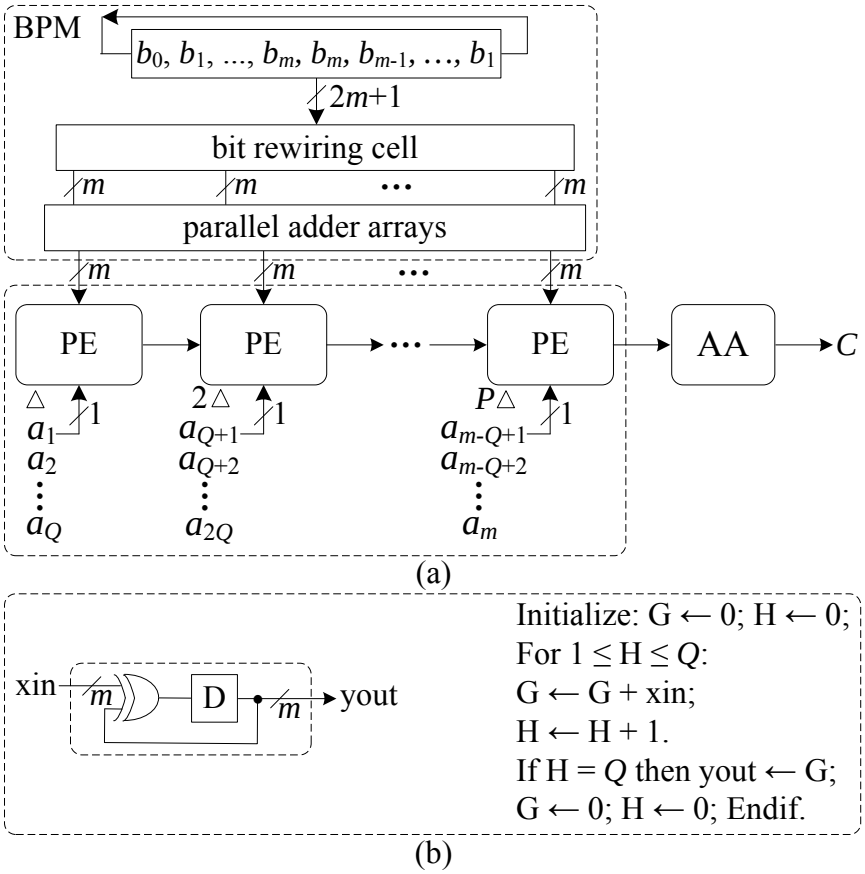

Figure 37: Proposed digit-serial (DS) architecture for RNB multiplier. (a) Proposed DS architecture. (b) Function of AA cell.

3. Final step

   3.1. $C = D$

---

where step 2.4 refers to the DS multiplication process. According to our proposed algorithm, we generate less number of partial products and partial products are accumulated as soon as they are computed, which not only shortens the ACT, but also significantly reduces register and adder complexity of proposed structures over that of existing ones in [79].

The proposed DS RNB multiplier based on Algorithm 6.3 is shown in Fig. 37(a), where it consists a BPM, P number of PEs and an addition-accumulation cell (AA). The bits

of operand $\overline{B}$ are firstly loaded in the circular shift-register, and then throughput the bit rewiring cell and parallel adder arrays to yield output to P PEs, which is exactly the same as that of array-1 in Fig. 35. The function of AA cell is described in Fig. 37(b), which consists of m parallel bit-level accumulation cells (consisting of an XOR gate and a bit-register). The newly received input is then added with the previously accumulated result and the result is stored in the register cell to be used during the next cycle. The accumulated output is obtained after Q cycles, and the duration of minimum cycle period is $(T_A + T_X)$. The proposed DS design gives the first output of desired product $(P+Q+1)$ cycles after the pair of operands are fed to the structure, while the successive output are produced at the interval of Q cycles thereafter. Similarly, we can extend the architecture in Fig. 37 further to obtain architectures with larger values of d, where P numbers of PEs are merged into P/d PEs as introduced in Section 6.3. Note that parallel adder array in BPM is based on MPB-I.

Many real-time applications, however, need DS architecture with various throughput rate to meet various area-time requirement. Thus, for any value of Q, let u and v be two integers such that $Q = uv + z$, where $0 \leq z < u$. For simplicity of discussion, we can assume $z = 0$ (and can append $(v - z)$ number of zeros to satisfy $Q = uv$, when $z \neq 0$). Define T as throughput rate of a structure for $1/Q \leq T < 1$. Then, we can rewrite (6.21) as

$$\begin{aligned} C = AB &= B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T \\ &= \underbrace{\sum_{i=0}^{v-1} B_i A_i^T + \sum_{i=v}^{2v-1} B_i A_i^T + \cdots + \sum_{i=Q-v-1}^{Q-1} B_i A_i^T}_{u \text{ terms}} \\ &= \sum_{j=0}^{u-1} \sum_{i=0}^{v-1} B_{i+jv} A_{i+jv}^T \end{aligned} \tag{6.23}$$

The architecture of Fig. 37 has throughput rate as $T = 1/Q$, i.e., can produce one product in every Q cycles. Based on (6.23), we can have the digit-serial algorithm for various throughput rate, i.e., $T = u/Q$, as stated in Algorithm 6.4.

Algorithm 6.4 Proposed digit-serial multiplication algorithm for throughput of $T = u/Q$

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

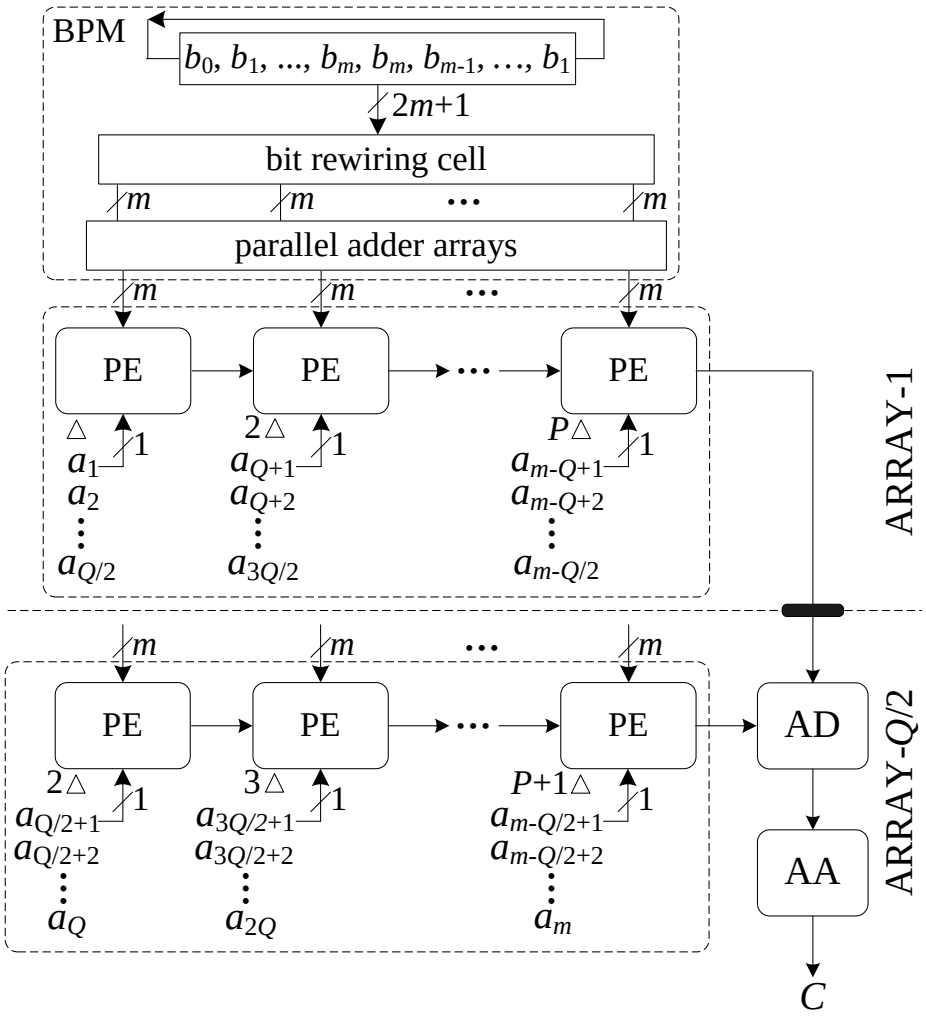

Figure 38: Proposed DS architecture for RNB multiplier for $T = 2/Q$.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, ..., $\overline{B}^{2m}$, $\overline{A}$

   1.2 $D = 0$

2. Multiplication step

   2.1. obtain $\overline{B}_S^1$, $\overline{B}_S^2$, ..., $\overline{B}_S^{2m+1}$, from $\overline{B}^1$, $\overline{B}^2$, ..., $\overline{B}^{2m}$ through bit-selecting operation

   2.2. for $j = 0$ to $u - 1$

2.3. for $i = 0$ to $v - 1$

2.4. for $h = 0$ to $P - 1$

2.5. $D = D + B_{i+jv} A^T_{i+jv}$

2.6. end for

2.7. end for

2.8. end for

3. Final step

3.1. $C = D$

To have higher throughput rate, e.g., $T = 2/Q$, we can have the structure as shown in Fig. 38, where double arrays of PEs (array-1 and array-$Q/2$ of Fig. 35) are used to achieve higher throughput rate compare with the former in Fig. 36. To meet the data processing requirement, the input bits to each PE of array-$Q/2$ are staggered by one cycle period, as shown in Fig. 38. After $(P+Q/2+2)$ cycles, the proposed architecture yields the first output and the successive output can be obtained in every $Q/2$ cycles. Likewise, the architecture of Fig. 38 can be extended to have architecture with other values of throughput rate ($T = u/Q$) and other values of d.

6.5 PROPOSED HYBRID RNB MULTIPLIER

The proposed multipliers presented in Sections 6.3 and 6.4 are very efficient in realization, but all these architectures can only meet one specific area/time requirement, i.e., for one specific architecture, it can not provide multiple choices of throughput rate. While for emerging computing platform, e.g., reconfigurable cryptographic system or resource constrained system, where one multiplier is required to meet various area-time requirement to achieve maximum resource utilization. Keeping these in view, we derive here a hybrid RNB multiplier with multiple throughput rate choices with small hardware overhead.

6.5.1 Algorithm

Ideally, the algorithm for the proposed hybrid RNB multiplier can be stated as follows:

Algorithm 6.5 Proposed hybrid multiplication algorithm

Inputs: A and B are the pair of elements (in RNB representation) in $GF(2^m)$ to be multiplied.

Output: $C = A \cdot B$

1. Initialization step

   1.1 define extended RNB $\{x_{m+1}, x_{m+2}, \ldots, x_{2m+1}\}$ to obtain $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$, $\overline{A}$

   1.2 $D = 0$

2. Multiplication step

   2.1. obtain $\overline{B}_S^1$, $\overline{B}_S^2$, …, $\overline{B}_S^{2m+1}$, from $\overline{B}^1$, $\overline{B}^2$, …, $\overline{B}^{2m}$ through bit-selecting operation

   If $T = 1/Q$

   Then:

   2.2. for $l = 0$ to $Q - 1$

   2.3. for $h = 0$ to $P - 1$

   2.4. $D = D + B_l A_l^T$

   2.5. end for

   2.6. end for

   2.7. $C = D$

   If $T = 2/Q$

   Then:

   2.2. for $j = 0$ to 1

   2.3. for $i = 0$ to $Q/2 - 1$

   2.4. for $h = 0$ to $P - 1$

   2.5. $D = D + B_{i+j(Q/2)} A_{i+j(Q/2)}^T$

   2.6. end for

   2.7. end for

2.8. end for

2.9. $C = D$

$\cdots \quad \cdots \quad \cdots$

If $T = Q/Q = 1$

Then:

2.2. for $i = 1$ to m

2.3. $C = B_0 A_0^T + B_1 A_1^T + \cdots + B_{Q-1} A_{Q-1}^T$

2.4. end for

6.5.2 Architecture

In this subsection, we give here the detailed design of proposed hybrid multiplier step by step. For simplicity of discussion, we only focus on the case of $d = 1$, though all these architectures can be extended further to have designs with larger values of d.

6.5.2.1 Architecture with two throughput choices The basic hybrid multiplier can have two throughput choices: $T = 1/Q$ and $T = 1$, i.e., it is the combination of architecture in Fig. 37 and Fig. 35. The proposed hybrid multiplier with two throughput choices is shown in Fig. 39. The architecture is nearly the same as that of Fig. 35 except a 1-to-2 demultiplexer (DM) cell and an AA cell. The function of DM cell is shown in Fig. 39(b), where S is the selector of throughput (S=0 refers to $T = 1$ and S=1 refers to $T = 1/Q$). The DM cell consists of m bit 1-to-2 demultiplexer (BDM) (the internal structure of BDM is shown in Fig. 39(b)) working in parallel (DM also contains $2m$ bit-registers to latch the output of m BDM during each cycle period, which are not shown in Fig. 39(b)).

In real applications, if we choose $T = 1/Q$, DM will connect the output of array-1 to the AA cell on its right to achieve DS realization (In this case, only array-1 is on the status of working, since input bits of operand A are serially loaded into the corresponding PEs according to Algorithm 5). After $(P + Q + 2)$ cycles (extra one cycle period is for DM cell), the AA cell produces the first result, and the successive one can be obtained in every

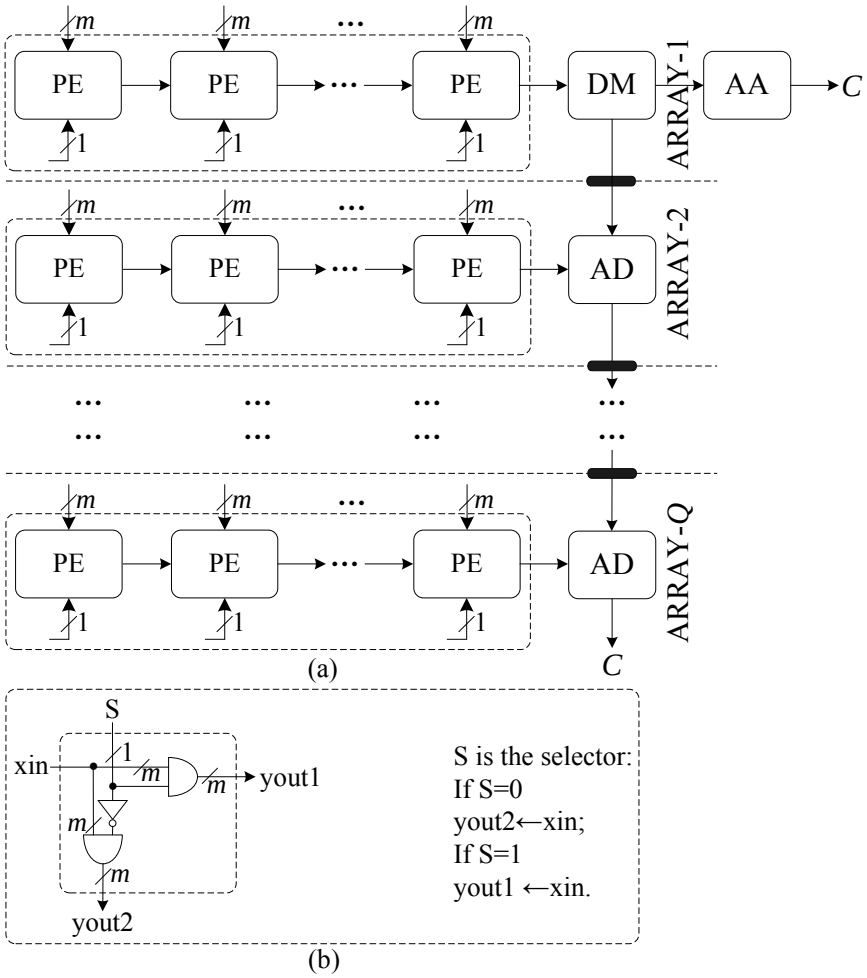

Figure 39: Proposed hybrid architecture for RNB multiplier with two throughput choices. (a) Proposed hybrid architecture. (b) Functional description of demultiplexer (DM) cell.

Q cycles. If we choose $T = 1$, DM cell will connect the output of array-1 to outputs of other arrays to achieve fully parallel realization (all the arrays will under working since the throughput rate is $T = 1$). After $(P + Q + 2)$ cycles (extra one cycle period is for DM cell), the last AD cell produces the first result, and the successive one can be obtained in every cycle thereafter. Note that the bits of operand A fed to each array (except array-1) will have one cycle delay compare to that of Fig. 35 to meet data path requirement.

Comparing to the architecture in Fig. 35, the proposed hybrid multiplier involves small hardware overhead: one extra DM cell and AA cell (though because of change of input style,

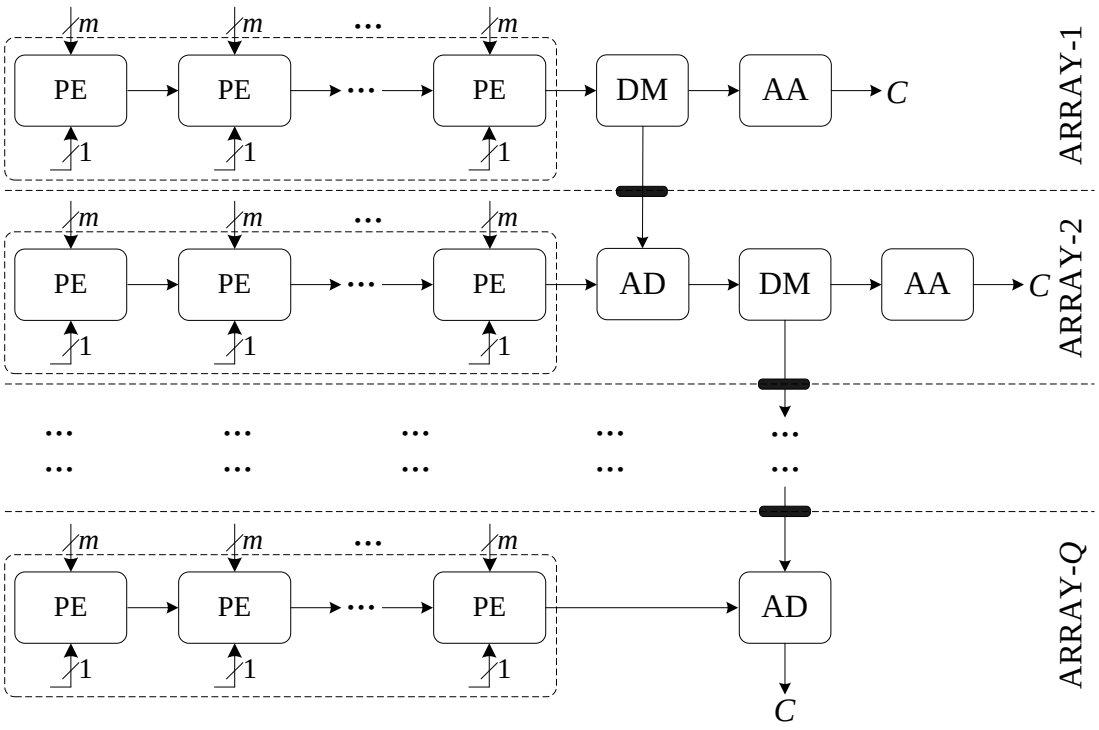

Figure 40: Proposed hybrid architecture for RNB multiplier with three throughput choices.

the BPM will be different, which is not discussed here). Overall, at the cost small area and time (one extra cycle) overhead, the hybrid multiplier can provide two throughput choices.

6.5.2.2 Architecture with three throughput choices Similarly, for three throughput choices: $T = 1/Q$, $T = 2/Q$ and $T = 1$, we can have proposed architecture as shown in Fig. 40.

The architecture of Fig. 40 has two extra DM cells and two extra AA cells to achieve three throughput capabilities. For choice of $T = 1/Q$, after a latency of $(P + Q + 2)$ cycles, the AA cell can have the first desired output word. For choice of $T = 2/Q$, the latency will be $(P + Q + 4)$ cycles, while for choice of $T = 1$, the latency for fist output is $(P + Q + 4)$ cycles. Likewise, each array (except array-1) will have certain cycles of delay to meet data processing requirement.

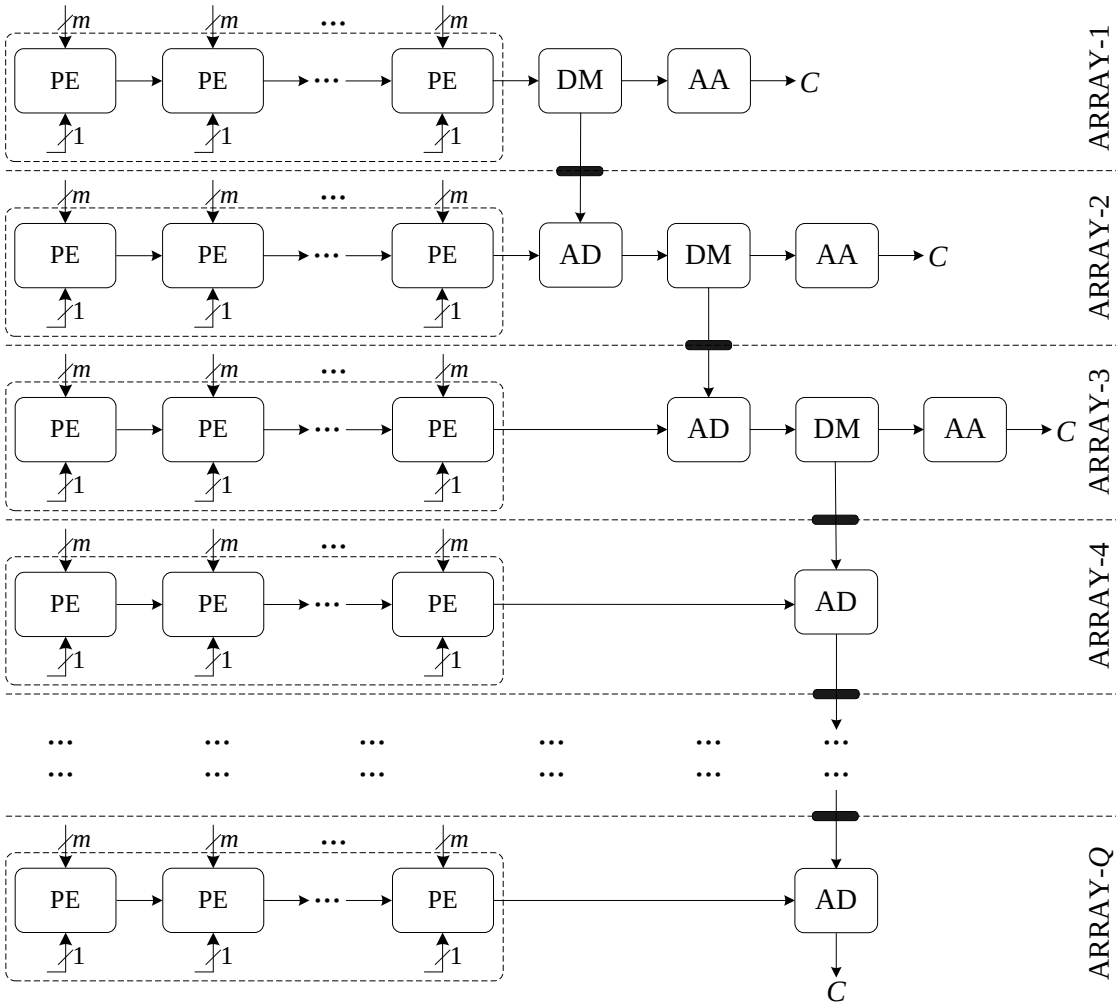

Figure 41: Proposed hybrid architecture for RNB multiplier with four throughput choices.

6.5.2.3 Architecture with four throughput choices The architecture with four throughput choices, i.e., $T = 1/Q$, $T = 2/Q$, $T = 3/Q$ and $T = 1$ is shown in Fig. 41, where involves extra three DM cells and three AA cells. The hybrid multiplier has latency of $(P + Q + 2)$ cycles, $(P + Q + 4)$ cycles, $(P + Q + 6)$ cycles and $(P + Q + 6)$ cycles for $T = 1/Q$, $T = 2/Q$, $T = 3/Q$ and $T = 1$, respectively.

Following the architectures of Figs. 39, 40 and 41, we can have a hybrid multiplier with $u + 1$ throughput choices, i.e., $T = 1/Q$, $T = 2/Q$, ..., $T = u/Q$ and $T = 1$, which requires extra u DM cells and u AA cells. For the choice of $T = u/Q$, the proposed hybrid multiplier

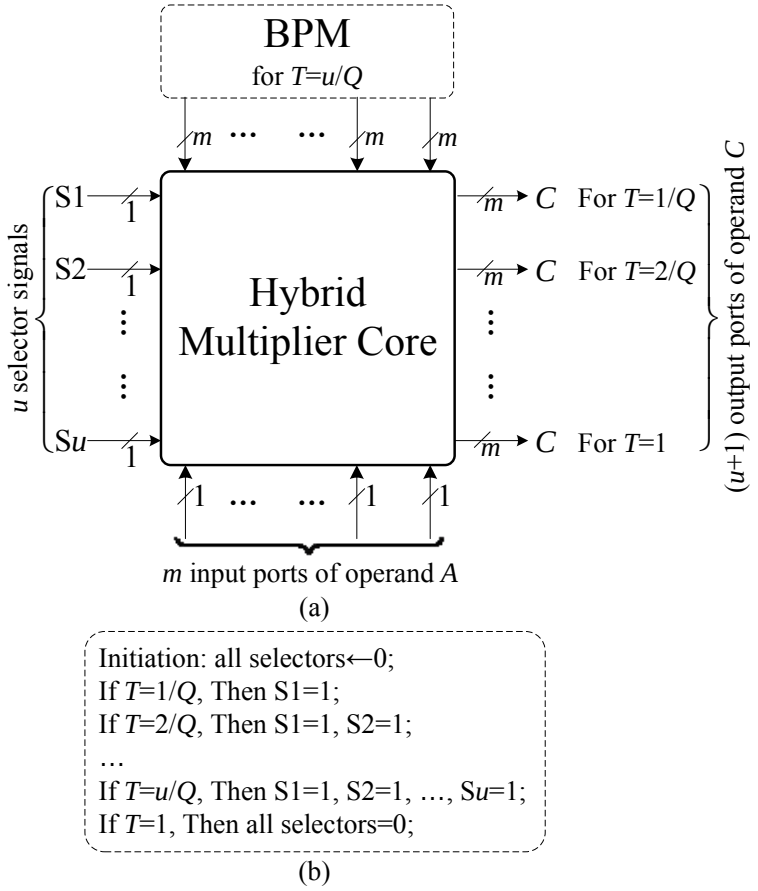

Figure 42: Design diagram for proposed hybrid RNB multiplier. (a) Proposed diagram. (b) Functional description of selector signals.

can yield its first output word after a latency of $(P + Q + 2u)$ cycles. Meanwhile, the input bits of operand fed to each PE in array-u should have $(2u - 2)$ cycles' delay to meet data transferring requirement. The proposed hybrid multiplier is highly modular and therefore can be applied in various application systems.

It is worth mentioning that for every throughput choice, we need a suitable BPM (the parallel adder arrays are fixed while the bit rewiring cell needs to be redesigned for each throughput choice) to meet the requirement of data processing. In real-time applications, we can have the major part of multiplier as a fixed multiplier core, while we can design individual BPM fit for each throughput choice (compare with the multiplier core, this part of area overhead is minor). The whole design diagram can be seen in Fig. 42 as well as the

functional description of selector signals. The proposed hybrid structure, therefore, can be used various application environments to meet specific requirements.

6.6 COMPLEXITY COMPARISON

6.6.1 Complexities of proposed MBP and DP architectures

Proposed MBP-I of Fig. 32 requires m PEs, where each of the $(m-1)$ regular PEs consists of m XOR gates and m AND gates, while the first PE (from left) requires only m AND gates. The proposed design in total requires $(2m^2 - m)$ XOR gates (parallel adder arrays require m^2 XOR gates) and m^2 AND gates. Besides, it needs a total of $(2m^2 + 2m)$ bit-registers, where m^2 registers are used for transferring data to the adjacent PE, $(m^2 + 2m)$ registers are used for registers in BPM. After a latency of $(m + 1)$ cycles, the proposed architecture gives the desired output word in every cycle of duration $(T_A + T_X)$.

MBP-II of Fig. 33 has similar area-complexity as MBP-I, i.e., MBP-II has m^2 AND gates, $(2m^2 - m)$ XOR gates and $(m^2 + 2m)$ bit-registers. Its critical-path is $(T_A + 2T_X)$, while the latency is m cycles.

For simplicity of discussion, the following architectures discussed only refer to those with BPM of MBP-I.

Proposed LBP architecture of Fig. 35 has the nearly same area-time complexities as that of Fig. 32 except the register count and latency (architecture of Fig. 35 has $(2m^2+m+Qm)$ bit-registers and has a latency of $(P + Q + 1)$ cycles). Note that when $P = Q = \sqrt{m}$, the architecture has shortest latency of $2\sqrt{m}$ cycles.

Proposed BP architecture for $d = 2$ is presented in Fig. 5. The technique used to reduce the register complexity for $d = 2$, however, can be extended further for any value of d, to obtain a structure of (m/d) PEs, where each of the regular PEs requires d AND cells and d XOR cells. The complete structure of the multiplier thus requires $(2m^2 - m)$ XOR gates, m^2 AND gates and $(m^2/d + m^2 + 2m)$ bit-registers. The latency of the structure amounts to $m/d + 1$ cycles, where the duration of minimum cycle period is $\{T_A + (1 + \lceil \log_2 d \rceil)T_X\}$.

Proposed LDP architecture with larger values of d, similarly, has $(2m^2 - m)$ XOR gates, m^2 AND gates and $(m^2/d + m^2 + m + Qm)$ registers. It has a latency of $(P/d + Q + 1)$ cycles with the critical path as $\{T_A + (1 + \lceil \log_2 d \rceil) T_X\}$.

6.6.2 Complexities of proposed DS architectures

Proposed DS architecture of Fig. 37 requires P PEs, where each of the $(P - 1)$ regular PEs consists of m XOR gates, m AND gates and m bit-registers. The first PE (from left) requires m AND gates and equal number of registers. Meanwhile, the AA cell requires m XOR gates and m registers. The proposed design in total would require $2Pm$ XOR gates (another Pm is used in parallel adder arrays) and Pm AND gates. Besides, it needs a total of $(2Pm + 3m)$ bit-registers, where Pm registers are used for transferring data to the adjacent PE, Pm registers are used in parallel adder arrays, m registers are used in the accumulator and another $2m$ registers are used for shift-registers in BPM. After a latency of $(P + Q + 1)$ cycles, proposed architecture gives the desired output word in every Q cycles of duration $(T_A + T_X)$.

Architecture of Fig. 37 with larger values of d requires $2Pm$ XOR gates, Pm AND gates and $(Pm/d + Pm + 3m)$ bit-registers. The latency of the structure amounts to $(P/d + Q + 1)$ cycles, where the duration of minimum cycle period is $\{T_A + (1 + \lceil \log_2 d \rceil) T_X\}$.

Proposed DS architecture with throughput $T = u/Q$ requires u arrays of PEs and $(u - 1)$ AD cells. In total, it requires $(Pum + Pm + um - m)$ XOR gates, Pum AND gates and $(Pum + Pm + um + 2m)$ registers. After a latency of $(P + Q + u)$ cycles, proposed architecture gives the desired output word in every Q/u cycles of duration $T_A + T_X$. If we apply the low-complexity technique with larger values of d, the proposed architecture in total requires the same XOR and AND gates as those of architecture for $d = 1$. Besides, it needs $(Pum/d + Pm + um + 2m)$ registers and a latency of $(P/d + Q + u)$ cycles with critical path of $\{T_A + (1 + \lceil \log_2 d \rceil) T_X\}$.

Table 9: Comparison of area-time complexities of proposed RNB multipliers

Design	AND	XOR	Register	Latency	Critical-path	ACT
Fig. 32	m^2	$2m^2-m$	$2m^2+2m$	$m+1$	T_A+T_X	T_{cp}
Fig. 33*	m^2	$2m^2-m$	m^2+2m	$m+1$	T_A+2T_X	T_{cp}
Γ^0	m^2	$2m^2-m$	$m^2/d+m^2+2m$	$m/d+1$	τ^0	T_{cp}
Fig. 35	m^2	$2m^2-m$	$2m^2+m+Qm$	$P+Q+1$	T_A+T_X	T_{cp}
Γ^1	m^2	$2m^2-m$	$m^2/d+m^2+m+Qm$	$P/d+Q+1$	τ^0	T_{cp}
Fig. 37	Pm	$2Pm$	$2Pm+3m$	$P+Q+1$	T_A+T_X	QT_{cp}
Γ^2	Pm	$2Pm$	$Pm/d+Pm+3m$	$P/d+Q+1$	τ^0	QT_{cp}
Γ^3	Pum	τ^1	$Pum+Pm+um+2m$	$P+Q+u$	T_A+T_X	QT_{cp}/u
Γ^4	Pum	τ^1	$Pum/d+Pm+um+2m$	$P/d+Q+u$	τ^0	QT_{cp}/u
Γ^5	m^2+2um	τ^2	$2m^2+3um+Qm+m$	Γ^7	T_A+T_X	Γ^9
Γ^6	m^2+2um	τ^2	$2m^2/d+3um+Qm+m$	Γ^8	τ^0	Γ^{10}

*: For simplicity of discussion, the following architectures listed only refer to those with BPM of MBP-I.

T_{cp}: Time duration of critical-path. ACT: Average computation time. $\tau^0 = T_A + (1+\lceil \log_2 d \rceil)T_X$ $\tau^1 = Pum + Pm + um - m$ $\tau^2 = 2m^2 + um - m$

Γ^0: Referring to BP architecture with larger values of d. Γ^1: Referring to LBP architecture with larger values of d. Γ^2: Referring to DS architecture with larger values of d. Γ^3: Referring to DS architecture with throughput $T = u/Q$. Γ^4: Referring to DS architecture with throughput $T = u/Q$ for larger values of d. Γ^5: Referring to hybrid architecture with with $(u+1)$ throughput choices. Γ^6: Referring to hybrid architecture with larger values of d. Γ^7, Γ^8: The latency of hybrid architecture depends on the throughput choice. Γ^9, Γ^{10}: The ACT of hybrid architecture depends on the throughput choice.

6.6.3 Complexity of proposed hybrid architecture

Proposed hybrid architecture with $(u + 1)$ throughput choices requires nearly the same area complexity as that of Fig. 35 except u extra DM cells and AA cells, where each DM cell requires $2m$ AND gates, m inverters and $2m$ registers and each AA cell needs m XOR gates and m registers. In total, the hybrid architecture (including the BPM) requires $(2m^2+um-m)$ XOR gates, (m^2+2um) AND gates, um inverters, and $(2m^2+3um+Qm+m)$ bit-registers. The critical path is the same as that of Fig. 35, i.e., $(T_A + T_X)$.

Proposed hybrid multiplier with larger values of d requires $(2m^2 + um - m)$ XOR gates, (m^2+2um) AND gates, um inverters, and $(2m^2/d+3um+Qm+m)$ bit-registers. Its critical path is $\{T_A + (1 + \lceil \log_2 d \rceil)T_X\}$.

6.6.4 Comparison of area and time complexities between proposed architectures

The area and time complexities, i.e., gate and register-counts, latency, critical-path, and average computation time (ACT) of all proposed multipliers are listed in Table 9 that different architecture has different area-time complexity. It is noted that when comparing with proposed LBP architecture of Fig. 35, proposed hybrid architecture involves only a little hardware overhead. Considering of its capability of providing multiple throughput rate choices, our proposed hybrid architecture can be used in various application environment.

6.6.5 Comparison with existing digit-serial multipliers

Since RNB is a reordered version of type II ONB, it is interesting to have comparison of proposed RNB multipliers with existing multipliers for the class of fields where having a type II ONB. The gate-counts, register-counts, latency, critical-path, and ACT of proposed digit-serial structures and existing structures of [68]-[70], [74], [77], [79]-[85] are listed in Table 10.

The authors in [79] and [82] have shown that their multipliers outperform the previously proposed structures in [68]-[70], [74], [77], [80]-[81], [83-85], respectively. Therefore we com-

Table 10: Comparison of area- and time-complexities of different digit-serial multipliers where there exist a type II ONB

Design	AND	XOR	Register	Latency	Critical-path	ACT
[77]	Pm	$2Pm - P$	$3m$	QT_{cp}	$T_A + (1 + \lceil \log_2(P+1) \rceil)T_X$	QT_{cp}
[80]	Pm	$3Pm/2 - P/2$	$3m$	QT_{cp}	$T_A + (1 + \lceil \log_2(P+1) \rceil)T_X$	QT_{cp}
[81]	Pm	$2Pm - 2P + m$	$2m$	QT_{cp}	$T_A + (2 + \lceil \log_2 P \rceil)T_X$	QT_{cp}
[82]	Pm	$2Pm - P/2 - P^2/2$	$2m$	QT_{cp}	$T_A + (1 + \lceil \log_2(P+1) \rceil)T_X$	QT_{cp}
[69]	τ^0	$2Pm - 2P$	$2m$	QT_{cp}	$T_A + \lceil \log_2(2m-1) \rceil T_X$	QT_{cp}
[74]	Pm	$2Pm - 2P$	$2m$	QT_{cp}	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	QT_{cp}
[83]^0	τ^1	$4Pm/3 - P^2 - 2P$	$2m$	QT_{cp}	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	QT_{cp}
[83]^1	τ^2	$2Pm - P^2 - 3P/2$	$2m$	QT_{cp}	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	QT_{cp}
[84]^2	τ^3	$2Pm - P$	$3m$	QT_{cp}	$2T_A + (3 + \lceil \log_2(P-1) \rceil)T_X$	QT_{cp}
[84]^3	τ^4	$Pm + Pm/2$	$3m$	QT_{cp}	$2T_A + (3 + \lceil \log_2(P-1) \rceil)T_X$	QT_{cp}
[85]^4	Pm	$2Pm - P^2/2 - 3P/2$	$3m$	QT_{cp}	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	QT_{cp}
[85]^5	Pm	$3Pm/2 - P/2$	$3m$	QT_{cp}	$T_A + (1 + \lceil \log_2(P+1) \rceil)T_X$	QT_{cp}
[68]	Pm	$2Pm - P$	$2m$	QT_{cp}	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	QT_{cp}
[70]	Pm	$2Pm$	$3m$	QT_{cp}	$T_A + (1 + \lceil \log_2(P+1) \rceil)T_X$	QT_{cp}
[79]-a[6]	$2Pm$	$4Pm - m$	τ^5	τ^8	$T_A + T_X$	τ^5
[79]-b[7]	Pm	$2Pm$	τ^6	τ^9	$T_A + 2T_X$	τ^6
Fig. 37*	Pm	$2Pm$	τ^6	τ^{10}	$T_A + 2T_X$	QT_{cp}
Fig. 37	Pm	$2Pm$	τ^7	τ^{10}	$T_A + T_X$	QT_{cp}

Table 11: Comparison of area- and time-complexities of different digit-serial multipliers where there exist a type II ONB (continual)

Design	AND	XOR	Register	Latency	Critical-path	ACT
[8]	Pm	$2Pm$	$(Pm/d + Pm + 3m)$	τ^8	$T_A + (1 + \lceil \log_2 d \rceil)T_X$	QT_{cp}
[9]	Pum	τ^{11}	$Pum + Pm + um + 2m$	τ^9	$T_A + T_X$	QT_{cp}/u
[10]	Pum	τ^{11}	$Pum/d + Pm + um + 2m$	τ^{10}	$T_A + (1 + \lceil \log_2 d \rceil)T_X$	QT_{cp}/u

*: For simplicity of discussion, the following architectures listed only refer to those with BPM of MBP-I. T_{cp}: Time duration of critical-path. ACT: Average computation time.

[0]: Referring to the AND-efficient digit-serial (AEDS). [1]: Referring to the XOR-efficient digit-serial (XEDS).

[2]: Referring to the ω-sequential multipliers with parallel output I (ω-SMPOI). [3]: Referring to the ω-SMPOII.

[4]: Referring to digit-level Gaussian normal basis multiplier with serial output (DLGM$_S$).

[5]: Referring to digit-level Gaussian normal basis multiplier with parallel output (DLGM$_P$).

[6]: Referring to word-level reordered normal basis multiplier type-a (WL-RNB-a). [7]: Referring to WL-RNB-b.

[8]: Referring to proposed DS architecture with larger values of d. [9]: Referring to proposed DS architecture for $T = u/Q$.

[10]: Referring to proposed DS architecture with larger values of d for $T = u/Q$.

τ^0: $\tau^0 = 2Pm - P$. τ^1: $\tau^1 = Pm - P^2/2 + P/2$. τ^2: $\tau^2 = 2Pm - P^2$. τ^3: $\tau^3 = Pm/2 + P + m$. τ^4: $\tau^4 = Pm + m$.

$\tau^5 = 2Pm + 2m + 2$ $\tau^6 = Pm + 2m + 1$ $\tau^7 = 2Pm + 3m + 1$

$\tau^8 = QT_{cp} + \lceil \log_2 2P \rceil T_X$. $\tau^9 = QT_{cp} + \lceil \log_2 P \rceil T_X$.

$\tau^{10} = P + Q + 1$. $\tau^{11} = Pum + Pm + um - m$ τ^8: $\tau^8 = P/d + Q + 1$. τ^9: $\tau^9 = P + Q + u$. τ^{10}: $\tau^{10} = P/d + Q + u$.

pare our proposed structures only with [79] and [82]. The proposed DS architecture (Fig. 37) has shorter ACT compared to the structure in [79] and [82]. Moreover, we have also proposed DS architectures with low register-complexity and high throughput rate, which are not covered in [79] and [82].

Note that for FPGA and ASIC implementation, we have chose $P = Q = \lceil \sqrt{m} \rceil$, since when $P = Q$, the latency of structure is the least among other choices of P and Q.

6.6.6 FPGA implementation

6.6.6.1 Optimal implementation FPGA based platform has been used in various computing system. However, it usually does not have abundant number of registers. We have proposed low register-complexity architectures, i.e., PEs in the arrays are merged together to reduce the register number, to enhance the FPGA implementation efficiency. While different series of FPGAs have different basic elements, here we just use one FPGA device to show the detailed process of achieving optimal realization (can be extended further to other FPGA devices).

We have used Altera Quartus II 12.0 and chosen Arria II GZ EP2AGZ225HF40C3 FPGA device to synthesize the proposed design (for simplicity of discussion, we just use the MBP architecture, though it can be used in other architectures) with various values of d ($d = 1,\ 2,\ 4,\ 8$). We have selected the field size of 233, where exists a type II ONB. After obtained the synthesis results, we have plotted the area-delay product (ADP) of MBP-I with various values of d, as shown in Fig. 43 (the number of ALUT is taken as the area measure, and the delay refers to the ACT). It can be seen that as d increases, ADP becomes more efficient. Due to the fact that one adaptive look-up table (ALUT) can contain multiple logic gates. In the following experiments, we decide to choose $d = 8$ as our default choice.

6.6.6.2 Comparison of FPGA implementation The key synthesis results of proposed designs (Fig. 37 and Fig. 37*) and existing designs ([79] and [82]) are obtained, in terms of area, maximum frequency and power consumption. Meanwhile, we have used the same input data and the same clock frequency (100MHz) to obtain the power synthe-

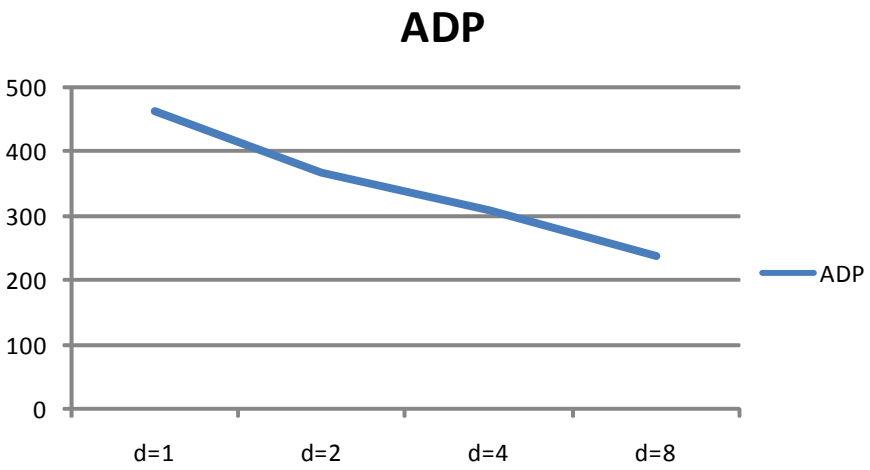

Figure 43: Area-delay product (ADP) trend along with d for proposed MBP-I. Unit: number of ALUT$\cdot 10^{-6}$.

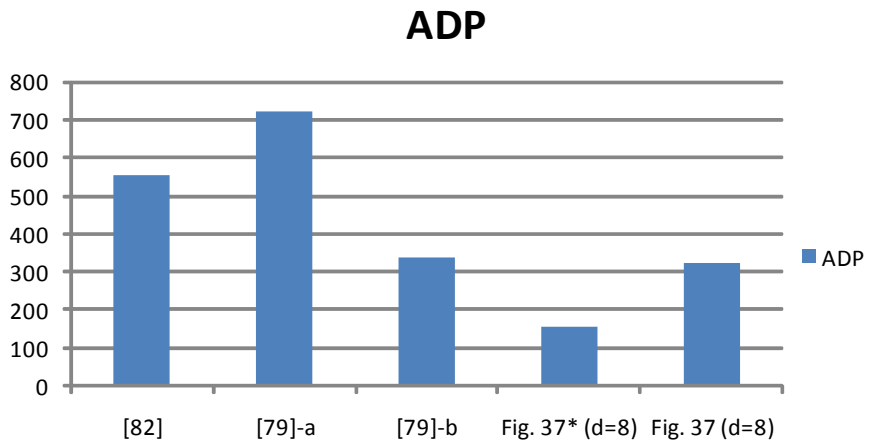

Figure 44: Area-delay product (ADP) comparison of proposed and existing structures. Unit: number of ALUT$\cdot 10^{-6}s$.

sis results using Quartus II PowerPlay Power Analyzer. We have also estimated the ADP, power-delay product (PDP) for various structures from the synthesis results are shown in Figs. 44 and 45, respectively.

It can be seen that for FPGA implementation, the proposed structures outperform the existing designs, for both ADP and PDP. It is worth noting that the ALUT of Altera FPGA devices can be mapped to logic function involving multiple Boolean operations, so that the

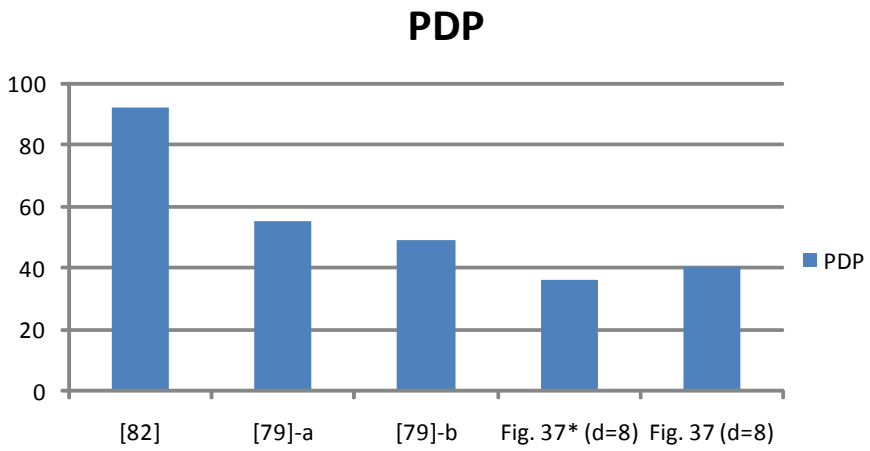

Figure 45: Power-delay product (PDP) comparison of proposed and existing structures. Unit: $\text{mW}\cdot 10^{-6}s$.

number of synthesized ALUT decreases as d increases. This technique is very useful in FPGA based designs.

6.6.7 ASIC implementation

6.6.7.1 Optimal implementation Similarly, we have also synthesized the proposed structure: MBP-I using Synopsys Design Compiler by North Carolina State University's 45nm FreePDK [55] to obtain the area and time complexities of the designs for various values of d. Using those synthesis results, we have plotted the ADP along with d in Fig. 46.

It can be seen that the performance of ADP decreases as d increases. This is due to the fact the ACT of the proposed structure increases with d. Thus, we choose $d = 1$ for default choice on ASIC implementation, though we can choose other values of d for implementation test.

6.6.7.2 Comparison of ASIC implementation We have also synthesized the proposed structures and the existing structures using Synopsys Design Compiler by North Carolina State University's 45nm FreePDK [55] to obtain the area, time and power complexities of the designs ($m = 233$). Using those synthesis results, we have obtained the area, and

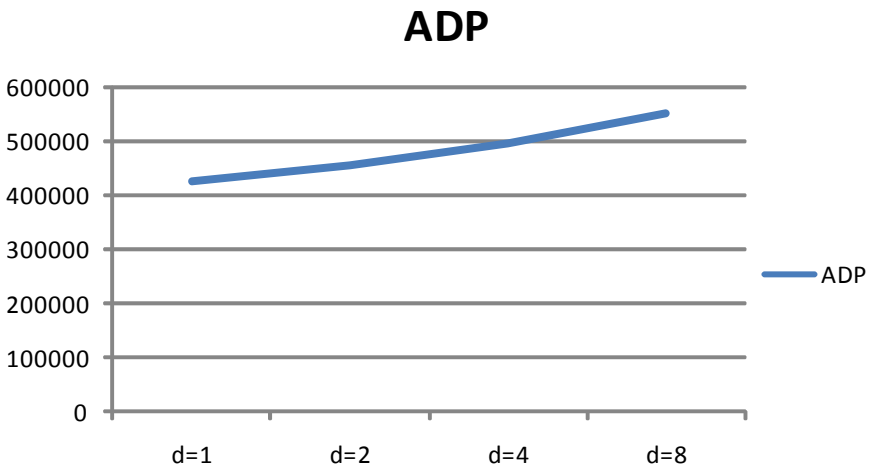

Figure 46: Area-delay product (ADP) trend along with d for proposed MBP-I. Unit: $u\text{m}^2\cdot\text{ns}$.

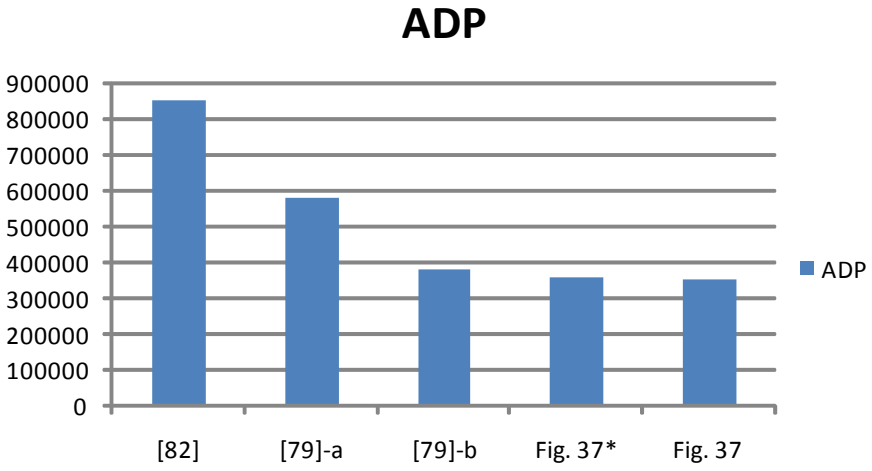

Figure 47: Area-delay product (ADP) comparison of proposed and existing structures. Unit: $u\text{m}^2\cdot\text{ns}$.

delay, and we have calculated the ADP of the designs (shown in Fig. 37), where proposed structures outperform the existing designs.

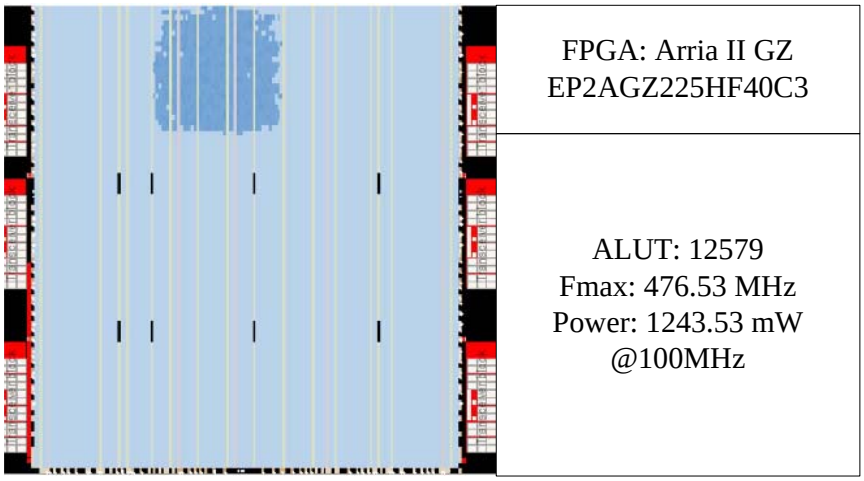

Figure 48: Chip-view of proposed hybrid architecture in FPGA device.

6.6.8 FPGA implementation of hybrid architecture

For the hybrid architecture proposed in Fig. 42, we have specifically implement it on FPGA platform, since its capability of providing various choices of throughput rate. Besides, it can also be used as an IP core for those reconfigurable cryptographic systems.

We have selected hybrid architecture with two throughput choices for $m = 233$, $P = Q = \sqrt{233}$ and $d = 8$. And we have synthesized the architecture and obtained the result. In total, proposed architecture has 12579 ALUT, maximum frequency is 476.53MHz and power consumption is 1243.54mW at 100MHz. We have also shown the detailed chip in Fig. 38.

6.7 CONCLUSION

Efficient bit- and digit-level algorithms for computing multiplication over $GF(2^m)$ based on RNB are proposed. high-throughput low-latency bit- and digit-parallel multipliers are presented first. Then, a novel digit-serial architecture for RNB multiplication is proposed. Finally, a novel hybrid architecture with various throughput rate choices is introduced for the first time with little hardware overhead. Both theoretical comparison and synthesis

simulation results from FPGA and ASIC implementation are presented. It is shown that the proposed high-throughput low-complexity multipliers have significantly lower area-time-power complexity than the corresponding existing designs. Specifically, FPGA realization of the novel hybrid multiplier is presented to confirm its potential application in FPGA based reconfigurable cryptographic platforms.

7.0 SUMMARY AND FUTURE WORK

7.1 DISSERTATION SUMMARY

In this dissertation, we have presented novel single and hybrid finite field multipliers over $GF(2^m)$ for emerging cryptographic systems. The following summarizes the contribution of this thesis.

In Chapter 3, which has been presented in [61], we have presented a novel PCA technique and modular reduction scheme for low-latency realization of bit-parallel Montgomery multiplication over $GF(2^m)$ based on irreducible pentanomials. We have decomposed the Montgomery multiplication into two concurrent blocks and we have further decomposed each block into several parallel processing arrays based on to proposed modular reduction scheme using PCA derive a lower- latency multiplier. The proposed design involves significantly less area-delay-power complexity than the newly reported multiplier for irreducible pentanomial, with at least one-fourth of the latency of the other, for the NIST recommended pentanomials.

In Chapter 4, which has been presented in [95], we have presented three pairs of low-latency high-throughput bit-parallel and digit-serial systolic structures for multipliers over $GF(2^m)$ based on NIST pentanomials. We have decomposed the multiplier into several parallel processing arrays to derive a pair of low latency bit-parallel and digit-serial systolic multipliers. Another pair of bit-parallel and digit-serial structures are then presented based on a novel modular reduction operation, where the critical-paths are smaller than those of the first pair ones. The third pair of designs, KA-based bit-parallel and digit-serial multipliers, are proposed to enhance the throughput rate further. The proposed designs, because of their flexibilities in latency choices, low area-time complexity and high throughput rate, can be used in various emerging cryptographic systems.

In Chapter 5, which has been presented in [96], we proposed three novel high-throughput digit-serial RB multipliers based on a novel recursive decomposition algorithm. We have suitably projected SFG of proposed algorithm and identified suitable cut-sets for feed-forward cut-set retiming to derive less area-time-power complexity implementation. Efficient structures with low register-count have been derived for area-constrained implementation; and particularly for implementation in FPGA platform where registers are not abundant. The results of synthesis show that proposed structures can achieve significant saving of ADPP for FPGA and ASIC implementation, respectively, over the best of the existing designs. The proposed structures have different area-time-power trade-off behavior, suitable for various application environments.

In Chapter 6, which has been presented in [97], we have proposed efficient bit- and digit-level algorithms for computing multiplication over $GF(2^m)$ based on RNB. We have firstly introduced high-throughput low-latency bit- and digit-parallel multipliers. Then, a novel digit-serial architecture for RNB multiplication is proposed. Finally, a novel hybrid architecture with various throughput rate choices is proposed for the first time with little hardware overhead. Both theoretical comparison and synthesis simulation results from FPGA and ASIC implementation are presented. It is shown that the proposed high-throughput low-complexity multipliers have significantly lower area-time- power complexity than the corresponding existing designs. Specifically, we have presented FPGA realization of the novel hybrid multiplier to confirm its potential application in FPGA based reconfigurable cryptographic systems.

Based on the above summary, the contributions of this dissertation are:

- Presenting low-latency bit-parallel Montgomery multiplier based on pentanomials.
- Introducing three pairs of low-latency (without any restriction) bit-parallel and digit-serial pentanomial basis multipliers
- Proposing three digit-serial RB multipliers with different characteristics for various application environments.
- For the first time, introducing several bit-parallel and digit-serial (single and hybrid) RNB multipliers for reconfigurable cryptographic systems.

7.2 FUTURE WORK

In the future, we would like to extend our research in the following aspects: (1) novel VLSI cryptographic systems for deeply embedded system security; (2) novel VLSI finite field arithmetic circuits.

- Novel VLSI cryptographic systems for deeply embedded system security. The focus of this theme of research includes design of hardware micro structure and specific platforms software. We will work on high-speed, power-efficient, lightweight, and small-size implementations of cryptographic systems that provide various security properties in various platforms, e.g., ASIC/FPGA and embedded processors, applicable to constrained, sensitive nodes in different applications ranging from industrial workstations to implantable and wearable medical devices. We will particularly explore solutions for extreme sensitive applications where crypto-measures are not feasible due to constraints in applications performance and implementation metrics, including low-power applications and wireless sensor networks. Moreover, we will focus on reliable approaches protecting various vulnerable implementations of cryptographic systems from attacking. At the same time, we will investigate to differentiate fault analysis attacks, VLSI defects, and denial-of-service attacks, considering a compromise between security and implementation metrics to realize the best system performance.
- Novel VLSI finite field arithmetic circuits. The focus of this theme of research includes design of hardware realization of finite field division, exponentiation and inversion. Although multiplication over finite field can also be transformed to these arithmetic operations, there are still great potential to realize these operations directly. Other methods of realization of finite field multiplication, e.g., Tom-Cook multiplication algorithm, will be applied to obtain more efficient realization.

BIBLIOGRAPHY

[1] S.-B. Wicker, and V.K. Bhargava, *Reed-Solomon Codes and Their Applications*, Piscatawy, NJ: IEEE Press, 1994.

[2] J.-P. Deschamps, J.L.Imana, and G.D. Sutter, *Hardware Implementation of Finite Field aArithmetic*. McGraw-Hill, 2009.

[3] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* Cambridge, U.K.: Cambridge Univ. Press, 1994.

[4] I. S. Reed andG. Solmon, "Polynomial Codes over Certain Finite Fields," *SIAM J. Appl. Math.*, 300-304, 1960.

[5] J. A. Solinas, "Efficient Arithmetic on Koblitz Curves," *Designs, Codes and Cryptography*, 19(1):195-249, 2000.

[6] D. F. Aranha, J.-L. Beuchat, J. Detrey, and N. Estibals, "Optimal Eta Pairing on Supersingular Genus-2 Binary Hyperelliptic Curves. In the cryptographers,"in *Track at the RSA Conference 2012 (CT-RSA 2012), LNCS*. New York, NY, USA: Springer, 98-115, 2012.

[7] J.-L. Beuchat, J. Detrey, N. Estibals, E. Okamoto, and F. Rodrguez- Henrquez, "Fast Architectures for The Pairing over Small-Characteristic Supersingular Elliptic Curves," *IEEE Trans. Comput.*, 60(2):266-281, 2011.

[8] NIST. Boulder, CO. Available: http://www.csrc.nist.gov/publications

[9] *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Std 1363-2000, Jan. 2000.

[10] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. Comput.*, 32(3):586-615, 2003.

[11] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *J. Cryptol.*, 17(4):297-319, 2004.

[12] L. H. Chen, P. L. Chang, C.-Y. Lee, and Y. K. Yang, "Scalable and Systolic Dual Basis Multiplier over," *Int. J. Innovative Computing, Inf. Contr.*, 7(3):1193-1208, Mar. 2011.

[13] A. H. Namin, H. Wu and M. Ahmadi, "A High-Speed Word Level Finite Field Multiplier in F_{2m} using Redundant Representation," *IEEE Trans. VLSI Systems*, 17(10):1546-1550, Oct. 2009.

[14] A. H. Namin, H. Wu and M. Ahmadi, "An Efficient Finite Field Multiplier using Redundant Representation," *ACM Trans. Embeded Comput. Sys.*, 11(2):Article 31, Jul., 2012.

[15] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over $GF(2^n)$,"in *Proc. CHES 2002, LNCS 2523*, 381-399, 2003.

[16] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata,"*Soviet Physics-Doklady* (English Translation), 7(7):595-596, 1963.

[17] I. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography, ser.* London Mathematical Society Lecture Note Series. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[18] [Online]. Available: http://cn.reuters.com/article/chinaNews/idCNCHINA-2987420100910

[19] [Online]. Available: http://en.wikipedia.org/wiki/Cryptography

[20] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* New York: Wiley, 1999.

[21] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[22] H. Fan, and M.A. Hasan, "Relationship Bbetween $GF(2^m)$ Montgomery and Shifted Polynomial Basis Multiplication Algorithms,"*IEEE Trans. Comput*, 55(9):1202-1206, 2006.

[23] C.-S. Yeh, I. S. Reed, and T. K. Truong, "Systolic Multipliers for Finite Fields $GF(2^m)$,"*IEEE Trans. Comput*, 33(4):357-360, Apr. 1984.

[24] Digital Signature Standard (DSS), FIPS 186-2, National Institute of Standards and Technology, 2000.

[25] P. K. Meher, "Systolic and Non-Systolic Scalable Modular Designs of Finite Field Multipliers for Reed-Solomon Codec,"*IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 17(6):747-757, June, 2009.

[26] C. H. Kim, C. P. Hong, and S. Kwon, "A Digit-Serial Multiplier for Finite Field $GF(2^m)$,"*IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, 13(4):476-483, 2005.

[27] N-Y. Kim, H-S. Kim, and K-Y. Yoo, "Computation of AB^2 Multiplication in $GF(2^m)$ using a Low-Complexity Systolic Architecture," *IEE Proc.-Circuits Devices Syst.*, 150(2):119-123, April, 2003.

[28] L. Song, K. K. Parhi, I. Kuroda, and T. Nishitani, "Hardware/Software Codesign of Finite Field Datapath for Low-Energy Reed-Solomn Codecs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst* 8(2):160-172, Apr. 2000.

[29] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Multicore Curve-Based Cryptoprocessor with Reconfigurable Modular Arithmetic Logic Units over $GF(2^n)$," *IEEE Trans. Comput.* 56(9):1269-1282, 2007.

[30] H. Wu, "Bit-Parallel Polynomial Basis Multiplier for New Classes of Finite Fields," *IEEE Trans. Comput*, 57(8):1023-1031, 2008.

[31] C. Paar, "Low Complexity Parallel Multipliers for Galois Fields $GF((2^n)^4)$ Based on Special Types of Primitive Polynomials", *IEEE International Symposium on Information Theory*, 1994.

[32] T.-C. Chen, S.-W. Wei, and H.-J. Tsai, "Arithmetic Unit for Finite Field $GF(2^m)$," *IEEE Circuits and System-I*, 55(3):828-837, 2008.

[33] N. R. Murthy, and M. N. S. Swamy, "Cryptographic Applications of Brahmaquptabha Skara Equation," *IEEE Circuits and System-I*, 53(7):1565-1571, 2006.

[34] C. Spagnol, E. M. Popovici, and W. P. Marnane, "Hardware Implementation of $GF(2^m)$ LDPC Decoder," *IEEE Circuits and System-I*, 56(12):2609-2620, 2009.

[35] H. Fan, and M.A. Hasan, "Relationship Between $GF(2^m)$ Montgomery and Shifted Polynomial Basis Multiplication Algorithms," *IEEE Trans. Computers*, 55(9):1202-1206, 2006.

[36] P. K. Meher, "On Efficient Implementation of Accumulation in Finite Field over $GF(2^m)$ and Its Applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 17(4):541-550, 2009.

[37] P. K. Meher, "Systolic and Super-Systolic Multipliers for Finite Field $GF(2^m)$ Based on Irreducible Trinomials," *IEEE Circuits and System-I*, 55(4):1031-1040, May, 2008.

[38] C.-Y. Lee, "Low Complexity Bit-Parallel Systolic Multiplier over $GF(2^m)$ using Irreducible Trinomials," *IEE Proc. Comput. Digit. Tech.*, 150(1):39-42, Jan. 2003.

[39] C.-Y. Lee, C.W. Chiou, J.-M. Lin and C.-C. Chang, "Scalable and Systolic Montgomery Multiplier over $GF(2^m)$ Generated by Trinomials," *IET Circuits Devices Syst.*, 1(6):477-484, 2007.

[40] B. Sunar and C. K. Koc, "Mastrovito Multiplier for All Trinomials," *IEEE Trans. Comput.*, 48(5):522-527, May 1999.

[41] W. Tang, H.Wu, and M. Ahmadi, "VLSI Implementation of Bit-Parallel Word-Serial Multiplier in $GF(2^{233})$," *in Proc. 3rd Int. IEEE-NEWCAS Conf.*, 399-402, Jun. 2005.

[42] T. Zhang and K. K. Parhi, "Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials," *IEEE Trans. Comput*, 50(7):734-749, Jul. 2001.

[43] Synposys, Design Ware. Foundry Libraries Mountain View, CA, 2005 [Online]. Available: http://www.synopsys.com/

[44] S.T.J. Fenn, M.G. Parker, M. Benaissa, and D. Taylo, "Bit-Serial Multiplication in $GF(2^m)$ using All-One Polynomials," *IEE proceedings in Computers and Digital Techniques*, 144(6):391-393, 1997.

[45] K.-Y. Chang, D. Hong, and H.-S. Cho, "Low Complexity Bit-Parallel Multiplier for $GF(2^m)$ Defined by All-One Polynomials using Redundant Representation," *IEEE Trans. Computers*, 54(12):1628-1629, 2005.

[46] H.-S. Kim, and S.-W. Lee, "LFSR Multipliers over $GF(2^m)$ Defined by All-One Polynomial," *Integration, the VLSI journal*, 40(4):571-578, 2007.

[47] P. K. Meher, Y. Ha, and C.-Y. Lee, "An Optimized Design of Serial-Parallel Finite Field Multiplier for $GF(2^m)$ Based on All-One Polynomials," *ASP-DAC* 2009, 210-215, 2009.

[48] C.-Y. Lee, E.-H. Lu, and J.-Y. Lee, "Bit-Parallel Systolic Multipliers for $GF(2^m)$ Fields Defined by All-One and Equally Spaced Polynomials," *IEEE Trans. Comput.*, 50(6):385-393, May, 2001.

[49] Y.-R. Ting, E.-H. Lu, and Y.-C. Lu, "Ringed Bit-Parallel Systolic Multipliers over a Class of Fields $GF(2^m)$," *Integration, the VLSI journal*, 38(4):571-578, 2005.

[50] C.-Y. Lee, J.-S. Horng, I.-C. Jou, and E.-H. Lu, "Low-Complexity Bit-Parallel Systolic Montgomery Multipliers for Special Classes of $GF(2^m)$," *IEEE Trans. Comput.*, 54(9):1061-1070, Sep. 2005.

[51] S. Talapatra, H. Rahaman, and J. Mathew, "Low Complexity Digit Serial Systolic Montgomery Multipliers for Special Class of $GF(2^m)$," *IEEE Trans. VLSI Sys.*, 18(5):847-852, May, 2010.

[52] T. Itoh and S. Tsujii, "Structure of Parallel Multipliers for a Class of Fields $GF(2^m)$," *Information and Computation*, 83(1):21-40, 1989.

[53] C.-Y. Lee, P.K. Meher, and C.-P. Chang, "Efficient M-ary Exponentiation over $GF(2^m)$ using Subquadratic KA-Based Three-Operand Montgomery Multiplier," *IEEE Trans. Circuits Systems-I.*, 2014.

[54] C.-L. Wang and J.-L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields $GF(2^m)$," *IEEE Trans. Circuits Syst.*, 38(7):796-800, Jul. 1991.

[55] North Carolina State University: 45nm FreePDK wiki, http://www.eda.ncsu.edu/wiki/FreePDK45:Manual

[56] C.W. Chiou, *et al.*, "Concurrent Error Detection in Montgomery Multiplication over $GF(2^m)$," *IEICE Trans. Fundamentals of Electronics, Comm. And Computer Sciences*, E89-A(2):566-574, 2006.

[57] C-Y. Lee, C-S. Yang, B.K. Meher, P.K. Meher, and J-S. Pan, "? Low-Complexity Digit-Serial and Scalable SPB/GPB Multipliers over Large Binary Extension Fields using (b,2)-Way Karatsuba Decomposition," *IEEE Trans. Circuits and Systems-I.*, 2014.

[58] S. K. Jain, L. Song, and K. K. Parhi, "Efficient Semisystolic Architectures for Finite Field Arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 6(1):734-749, Mar. 1998.

[59] S. B.-Sarmadi, and M.A. Hasan, "Concurrent Error Detection in Finite Field Arithmetic Operations Using Pipelined and Systolic Architectures," *IEEE Trans. Comput.*, 58(11):1553C1567, Nov. 2009.

[60] A. Hariri, and A. Reyhani-Masoleh, "Digit-Level Semi-Systolic and Systolic Structures for the Shifted Polynomial Basis Multiplication Over Binary Extension Fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 19(11):2125-2129, Mar. 2009.

[61] J. Xie, J. He, and P. K. Meher, "Low Latency Systolic Montgomery Multiplier for Finite Field $GF(2^m)$ Based on Pentanomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, 21(2):385-389, Feb. 2013.

[62] S. Kumar, T. Wollinger, and C. Paar, "Optimum Digit Serial $GF(2^m)$ Multipliers for Curve-Based Cryptography," *IEEE Trans. Computers*, 55(10):1306-1311, Oct. 2006.

[63] C. H. Kim, C. P. Hong, and S. Kwon, "A Digit-Serial Multiplier for Finite Field $GF(2^m)$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 13(4):476-483, Apr. 2005.

[64] J-S. Pan, C-Y. Lee, and P. K. Meher, "Low-Latency Digit-Serial and Digit-Parallel Systolic Multipliers for Large Binary Extension Fields," *IEEE Trans. on Circuits and Systems-I: Regular Papers*, 60(12):3195-3204, Dec. 2013

[65] P. L. Montgomery, "Five, Six, and Seven-Term Karatsuba-Like Formulae," *IEEE Trans. Computers*, 54(3):362-369, 2006.

[66] NanGate Standard Cell Library [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib/

[67] G. Drolet, "A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic Circuits," *IEEE Trans. Comput.*, 47(9):938-946, 1998.

[68] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao, "Finite Field Multiplier using Redundant Representation," *IEEE Trans. Comput.*, 51(11):1306-1316, Nov. 2002.

[69] J.L. Massey and J.K. Omura, "Computational Method and Apparatus for Finite Field Arithmetic,"US Patent 4587627, 1984.

[70] A. H. Namin, H. Wu and M. Ahmadi, "Comb Architectures for Finite Field Multiplication in F_{2m}," *IEEE Trans. Comput.*, 56(7):909-916, Jul. 2007.

[71] A. H. Namin, H. Wu and M. Ahmadi, "A New Finite Field Multiplier using Redundat Representation," *IEEE Trans. Comput.*, 57(5):716-720, May 2008.

[72] S. Talapatra, H. Rahaman, and S. K. Saha, "Unified Digit Serial Systolic Montgomery Multiplication Architecture for Special Classes of Polynomials over $GF2^m$,"in *Proc. Euromicro Conf. Digital System Design: Architectures, Meth. Tools*, 427-432, 2010.

[73] J. Rajski and J. Tyszer, "Primitive Polynomials over of Degree up to 660 with Uniformly Distributed Coefficients," *J. Electronic Testing: Theory Appl.*, 19:645-657, 2003.

[74] S. Gao and S. Vanstone, "On Orders of Optimal Normal Basis Generators," *Math. Comput.*, 64(2):1227-1233, 1995.

[75] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over $GF(2^m)$," *IEEE Trans. Computers*, 52(4):428-439, Apr. 2003.

[76] A. Reyhani-Masoleh and M. A. Hasan, "Low Complexity Word-Level Sequential Normal Basis Multipliers," *IEEE Trans. Comput.*, 54(2):98-110, Feb. 2005.

[77] A. H. Namin, H. Wu and M. Ahmadi, "A Word-Level Finite Field Multiplier using Normal Basis," *IEEE Trans. Comput.*, 60(6):890-895, Jun. 2011.

[78] A. Hariri, and A.R.-Masoleh, "Bit-Serial and Bit-Parallel Montgomery Multiplication and Squaring over $GF(2^m)$,"*IEEE Trans. Comput.*, 58(10):1332-1345, Oct. 2009.

[79] A. H. Namin, H. Wu and M. Ahmadi, "High-Speed Architectures for Multiplication Using Reordered Normal Basis," *IEEE Trans. Comput.*, 61(2):164-172, Feb. 2012.

[80] R. Azarderakhsh, and A. Reyhani-Masoleh, "A Modified Low Complexity Digit-Level Gaussian Normal Basis Multiplier," *Proc. Third Int'l Workshop Arithmetic of Finite Fields (WAIFI)*, 25-40, Jun. 2010.

[81] C.-Y. Lee, "Concurrent Error Detection Architectures for Gaussian Normal Basis Multiplication over $GF(2^m)$," *Integration, the VLSI J.*, 43(1):113-123, 2010.

[82] R. Azarderakhsh, and A. Reyhani-Masoleh, "Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases," *IEEE Trans. Comput.*, 62(4):744-757, Apr. 2013.

[83] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over $GF(2^m)$," *IEEE Trans. Computers*, Special Issue on Cryptographic Hardware and Embedded Systems, 52(4):428-439, Apr. 2003.

[84] A. Reyhani-Masoleh and M. A. Hasan, "Low Complexity Word-Level Sequential Normal Basis Multipliers," *IEEE Trans. Comput.*, 54(2):98-110, Feb. 2005.

[85] A. Reyhani-Masoleh, "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," *IEEE Trans. Comput.*, 55(1):34-47, Jan. 2006.

[86] Jiafeng Xie, Jianjun He and P.K. Meher, "Hardware-Efficient Realization of Prime-Length DCT Based on Distributed Arithmetic" *IEEE Trans. Computers*, 62(6):1170-1178, 2013.

[87] Jiafeng Xie, P.K. Meher and Jianjun He, "Low-Complexity Multiplier for $GF(2^m)$ Based on All One Polynomials" *IEEE Trans. on VLSI Systems*, 21(1):168-172, 2013.

[88] Jiafeng Xie, Jianjun He, and Guanzheng Tan, "FPGA Realization of FIR Filters for High-speed and Medium-speed by Using Modified Distributed Arithmetic Architectures" *Microelectronics Journal* (Elsevier), 41(6):365-370, 2010.

[89] Jiafeng Xie, Jianjun He and Weihua Gui, "Low Latency Systolic Multipliers for Finite Field $GF(2^m)$ Based on Irreducible Polynomials" *Journal of Central South University of Technology*, 21(5):1283-1289, 2012.

[90] Jianjun He, Jiafeng Xie, and Mingfang He, "Area-efficient Systolic Multipliers for Finite Field $GF(2^m)$ Based on Irreducible Trinomial" *Journal of Convergence Information Technology*, 6(5):305-313, 2011.

[91] Jianjun He, and Jiafeng Xie, "Hardware Efficient Approach for Memoryless-Based Multiplication and Its Application to FIR Filter" *Journal of Computers*, 6(11):2376-2381, 2011.

[92] Jiafeng Xie, and Guanzheng Tan, "Design of B-mode Ultrasonic Imaging System Based on FPGA" *Journal of Zhongyuan University of Technology*, 21(2):72-75, 2010.

[93] Jiafeng Xie, and Guanzheng Tan, "The Research on Locating the Car License in the Static Image Based on MATLAB" *PLC&FA*, 6(1):668-671, 2009.

[94] Jiafeng Xie, P.K. Meher and Jianjun He, "Low-Latency Area-Delay-Efficient Systolic Multiplier over $GF(2^m)$ for a Wider Class of Trinomials using Parallel Register Sharing" *IEEE International Symposium on Circuits and Systems-2012, ISCAS-12*, 89-92, 2012

[95] Jiafeng Xie, P.K. Meher and Zhi-Hong Mao, "Low-Latency High-Throughput Systolic Multipliers over $GF(2^m)$ for NIST Pentanomials,"accepted in *IEEE Trans. Circuits and Systems-I*

[96] Jiafeng Xie, P.K. Meher and Zhi-Hong Mao, "High-Throughput Finite Field Multipliers Using Redundant Basis for FPGA and ASIC Implementations,"accepted in *IEEE Trans. Circuits and Systems-I*

[97] Jiafeng Xie, P.K. Meher and Zhi-Hong Mao, "Single and Hybrid Architectures for Multiplication over Finite Field $GF(2^m)$ Based on Reordered Normal Basis,"to be submitted to *IEEE Trans. Computers*

[98] Jiafeng Xie, P.K. Meher and Zhi-Hong Mao, "High-Throughput Digit-Level Systolic Multipliers over $GF(2^m)$ Based on Irreducible Trinomials,"accepted in *IEEE Trans. Circuits and Systems-II*

[99] Jiafeng Xie, P.K. Meher and Zhi-Hong Mao, "Efficient Realization of All-One Polynomial Basis Multiplier and Its Applications,"to be submitted to *IEEE Trans. VLSI Systems*