2016

FPGA Realization of Low Register Systolic Multipliers over GF(2^m)

Qiliang Shao
Wright State University

FPGA REALIZATION OF LOW REGISTER SYSTOLIC MULTIPLIERS OVER $GF(2^m)$

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

By

QILIANG SHAO
B.E., Donghua University, China, 2014

2016

Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

December 8, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Qiliang Shao ENTITLED FPGA REALIZATION OF LOW REGISTER SYSTOLIC MULTIPLIERS OVER $GF(2^m)$. BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Electrical Engineering.

Jiafeng Xie, Ph.D.
Thesis Director

Brian Rigling, Ph.D.
Chair, Electrical Engineering

Committee on Final Examination

Jiafeng Xie, Ph.D.

Yan Zhuang, Ph.D.

Zhiqiang Wu, Ph.D.

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

Abstract

Qiliang,Shao. M.S.E.E., Department of Electrical Engineering, Wright State University, 2016. FPGA realization of low register systolic multipliers over $GF(2^m)$.

Finite field multiplication over $GF(2^m)$ is a critical component for elliptic curve cryptography (ECC). National Institute of Standards and Technology (NIST) has recommended five polynomials (two trinomials and three pentanomials) for ECC implementation. Although there are a lot reports available on polynomial basis multipliers, efficient implementation of a design with flexible field-size is quite rare. There is another basis to represent the field called normal basis. Normal basis multiplication over $GF(2^m)$ is widely used in various applications such as elliptic curve cryptography (ECC). As a special class of normal basis with low complexity, Gaussian normal basis (GNB) has received considerable attention recently. In this paper, we first propose a novel low-complexity hybrid-size systolic polynomial basis multiplier based on a proposed novel hybrid-size (for both pentanomial and trinomial) algorithm for efficient systolization of finite field multiplications. Next, we propose a novel decomposition algorithm to develop a digit-level (DL) low critical-path delay and low register-complexity systolic structure for GNB multiplication over $GF(2^m)$. For the hybrid-size systolic polynomial multipliers, both the theoretical and field-programmable gate array (FPGA) implementation show that, our proposed architectures have lower register-complexity than the existing ones. The proposed hybrid-size multiplier can also be extended to other field-size and can be used as a third-party intellectual property (IP) core for various cryptosystems. At the same time, the proposed systolic Gaussian normal basis multipliers can achieve both low critical-path and low register-complexity through the theoretical and application-specific integrated circuit (ASIC) comparisons with the existing GNB multipliers.

Key Words-Gaussian normal basis (GNB), finite field multiplication, systolic structure, irreducible polynomials, low complexity, digit-level, low critical-path delay.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter will introduce the outline of the whole thesis. It presents some existing works of finite field multipliers, both polynomial multipliers and GNB multipliers. The contributions of this report are also given here.

1.1 Preliminary

1.1.1 Polynomial Multipliers

Elliptic curve cryptography (ECC) is widely used in many fields such as wearable devices and portable systems [1-4]. Finite field multiplication over $GF(2^m)$ is a crucial part of ECC, and there are mainly two bases selected to represent the field operation, i.e., polynomial basis [5-13] and normal basis [14-17]. Due to the simpler design structures, polynomial-based multipliers are more popularly used in hardware implementation than normal basis ones [8].

In general, pentanomials and trinomials are two main irreducible polynomials [7-11], [17-26]. The National Institute of Standards and Technology (NIST) has recommended three pentanomials and two trinomials, which are popularly used for ECC implementation [5]. Although there are many reports available in the literature focusing efficient implementation of finite field multipliers based one either pentanomial or trinomial, there is still few specific design with hybrid-size multiplier.

Systolic design and non-systolic design are two basic structures for field multipliers over $GF(2^m)$. Because of the modularity and regularity of the structure, polynomial-based systolic multipliers are considered to be used in applications [5-11]. Also, systolic design

has the pipelining structure, and the registers must be used in all the processing elements (PEs) in the systolic array [5]. Therefore, systolic structures usually have higher register-complexity, compared with lower complexity and longer time delay of non-systolic designs. Systolic structure with low register-complexity are needed when realized in field-programmable gate array (FPGA) platforms, where the register-resources are not so rich. Based on the irreducible polynomials, a great deal of efforts have been made to reduce the complexity [7-10], [23-26]. In [23], Lee *et al.* introduced a bit-parallel AOP-based systolic multiplier. In addition, Xie *et al.* [24] presented another efficient AOP-based structure of multiplier. In [7], a bit-parallel trinomial based systolic multiplier has been presented by Lee *et al.* Meher [8] introduced efficient bit-parallel systolic and super-systolic designs. A low register-complexity systolic structure has been proposed in [9]. Jos Luis Imana *et al.* [28] introduced a low-complexity bit-parallel multiplier based on irreducible pentanomials. Many other works were reported for implementation of finite field multiplication over $GF(2^m)$ [11], [17].

1.1.2 Gaussian Normal Basis Multipliers

Low-complexity implementations of finite field multipliers over $GF(2^m)$ have drawn substantial attention recently due to their widespread applications in various environments. A lot of efforts have been carried to obtain low-complexity multipliers with high-performance for various usage including elliptic curve cryptography (ECC) [1], [31]-[33].

In general, there are three bases can be selected to represent a finite field, i.e., polynomial basis, normal basis, and dual basis [34]-[48]. Compared with polynomial basis and dual basis, normal basis is much more efficient in the hardware designs involving with many squaring operations since normal basis has an advantage that the squaring of field elements can be simply obtained by cyclic shifting without hardware usage. Compared with other two bases, Normal basis, therefore, has drawn much more attention in the applications which utilize frequent squarings.

Gaussian normal basis (GNB), as a special class of normal basis over $GF(2^m)$ [34]-[38] (where $m > 1$ and m is not divisible by eight), has received considerable attention in the literature due to its low complexity. GNB has been included in a number of standards such as IEEE [49] and NIST [50] for elliptic curve digital signature algorithm (ECDSA). According to the structuring of finite field multipliers based on normal basis, especially

GNB, the respective implementations can be categorized into three groups: 1) bit-level including parallel-in serial-out (PISO) [51], serial-in parallel-out (SIPO) [52]-[53], and parallel-in parallel-out (PIPO) [54]-[55]; 2) digit-level which includes the structures of: parallel-in serial-out [56], parallel-in parallel-out [46]-[47], and serial-in parallel-out [48]; and 3) bit-parallel, which includes the multiplier of [32].

For large field sizes in $GF(2^m)$, the multiplications can be realized by using systolic array to achieve high-speed and regular implementations [39]. Systolic structures are vastly used in applications with high-performance requirements as the processing elements (PEs) in the structure employ registers for pipelining. In [40], Kwon has presented an efficient digit-serial systolic multiplier based on optimal normal basis. In [41], another systolic multiplier is proposed for high-performance implementation. Other efficient systolic multipliers over $GF(2^m)$ have been proposed in [39] and [42]. Besides that, efficient digit-serial systolic multipliers are introduced in [26], [43]-[44]. Very recently, an efficient digit-level (DL) systolic GNB multiplier is introduced in [48]. Although these GNB multipliers have been optimized to achieve low complexity through DL implementation, their area-time complexities are still relatively high and need to be improved.

In Chapter 3 of this thesis, we propose a novel decomposition algorithm to develop DL systolic GNB multipliers over $GF(2^m)$ in order to achieve low critical-path delay, high-speed and low-complexity implementations. First, we introduce novel multiplication algorithms to reduce the respective critical-path delay and register-complexity. Then, a new structure of proposed systolic GNB multiplier is proposed. The proposed multiplier can achieve low critical-path delay and low register-complexity compared with the best of the existing GNB systolic multiplier of [48]. Finally, we also compare the hardware and time complexities of the proposed architectures with the existing ones through application-specific integrated circuit (ASIC) synthesis to benchmark the higher efficiencies of the presented design.

1.2 Summery of contribution

1.2.1 Proposed Low-Complexity Hybrid-Size Systolic Polynomial Multipliers over $GF(2^m)$

Many designs about finite field polynomial systolic multipliers have been reported. Most of these designs focus on the specific NIST recommended polynomials, such as pentanomials and trinomials. In Chapter 2 of this thesis, we propose a novel hybrid-size (for both pentanomial and trinomial) algorithm for efficient systolization of finite field multiplications. After that, an detailed example of a hybrid-size multiplier (combined with $GF(2^{163})$ and $GF(2^{233})$) is presented as well as the proposed systolic structure. Both the theoretical and field-programmable gate array (FPGA) implementation show that, the proposed hybrid-size design (can perform both $GF(2^{163})$ and $GF(2^{233})$ multiplications) are found to have at least 70.0% and 47.6% less area-delay product (ADP) and power-delay product (PDP) than the combination of proposed individual $GF(2^{163})$ and $GF(2^{233})$ multipliers (best of all existing designs), respectively. Besides that, the proposed hybrid-size one only involves 70.0% and 47.6% more ADP and PDP than the proposed individual $GF(2^{233})$ multiplier, respectively.

1.2.2 Proposed Low-Complexity Systolic Gaussian Normal Basis Multipliers over $GF(2^m)$

A lot of works have been done concerning Gaussian normal basis multipliers. In Chapter 3 of this thesis, we propose a novel decomposition algorithm to develop a digit-level (DL) low critical-path delay and low register-complexity systolic structure for GNB multiplication over $GF(2^m)$. Compared with the existing digit-level GNB multipliers (through both the theoretical and application-specific integrated circuit (ASIC) comparison), the proposed multiplier not only has lower critical-path delay, but also achieves significantly less area-delay product (ADP), e.g., for a systolic structure of digit-size of 7 for $GF(2^{409})$, the proposed structure has 28.9% less critical-path delay and 26.8% less ADP compared to the best of the existing designs in [24], respectively.

1.2.3 Report Outline

The following parts of the report are organized in this way.

Chapter 2 proposes hybrid-size systolic polynomial multipliers over $GF(2^m)$ with low complexity. Several classes of irreducible polynomials are shown within this chapter.

Chapter 3 talks about the low critical-path low-complexity digit-level systolic gaussian normal basis multipliers.

Chapter 4 presents the conclusion for the whole thesis.

Chapter 2

Efficient Implementation of Low Complexity Hybrid-Size Systolic Polynomial Multipliers over $GF(2^m)$

2.1 Low Register-Complexity Systolic Multipliers Based on NIST Trinomials and Pentanomials

In this section, we briefly review the conventional algorithm and existing systolic structures for NIST trinomials and pentanomials first, and then present the modified low register-complexity systolic structures.

2.1.1 Review of Conventional Polynomial Multiplication

Let's define $F(t) = t^m + p_{m-1} \cdot t^{m-1} + \cdots + p_2 \cdot t^2 + p_1 \cdot t + 1$, as an irreducible polynomial of degree m over $GF(2^m)$ where $p_i \in GF(2)$ are the coefficients. The polynomial basis $\{1, t, t^2, \ldots, t^{m-1}\}$, where t is a root of $F(t)$, is used to represent the field elements. Therefore, the three elements $A, B, C \in GF(2^m)$ can be represented as

$$A = \sum_{i=0}^{m-1} a_i \cdot t^i,\ B = \sum_{i=0}^{m-1} b_i \cdot t^i,\ C = \sum_{i=0}^{m-1} c_i \cdot t^i, \tag{2.1}$$

where $a_i, b_i, c_i \in GF(2)$, for $0 \leq i \leq m-1$.

Let us define C as the product of A and B, then we have

$$C = A \cdot B \bmod F(t). \tag{2.2}$$

The above equation (2.2) can also be written in the form of:

$$C = \sum_{i=0}^{m-1} b_i \cdot (t^i \cdot A \bmod F(t)). \tag{2.3}$$

Let us define

$$\begin{aligned} A^{(0)} &= A \\ A^{(1)} &= t^i \cdot A \bmod F(t), \end{aligned} \tag{2.4}$$

so that we can derive $A^{(i+1)}$ from $A^{(i)}$ recursively that

$$A^{(i+1)} = t \cdot A^{(i)} \bmod F(t). \tag{2.5}$$

Then, we have

$$A^{(i+1)} = (a_0^i \cdot t + a_1^i \cdot t^2 + \cdots + a_{m-1}^i \cdot t^m) \bmod F(t), \tag{2.6}$$

where

$$A^{(i)} = \sum_{j=0}^{m-1} a_j^i \cdot t^j. \tag{2.7}$$

Because t is a root of polynomial $F(t)$, then we have

$$F(t) = t^m + p_{m-1} \cdot t^{m-1} + \cdots + p_1 \cdot t + 1 = 0, \tag{2.8}$$

then we can have

$$t^m = p_{m-1} \cdot t^{m-1} + \cdots + p_1 \cdot t + 1. \tag{2.9}$$

Substituting the t^m into (2.7), then we have

$$\begin{aligned} A^{(i+1)} &= a_{m-1}^i + (a_0^i + p_1 \cdot a_{m-1}^i) \cdot t \\ &+ \cdots + (a_{m-2}^i + p_{m-1} \cdot a_{m-1}^i) \cdot t^{m-1}. \end{aligned} \tag{2.10}$$

Let's define

$$A^{(i+1)} = a_0^{i+1} + a_1^{i+1} \cdot t + \cdots + a_{m-1}^{i+1} \cdot t^{i+1}, \tag{2.11}$$

then we can have

$$\begin{aligned} a_0^{i+1} &= a_{m-1}^i \\ a_j^{i+1} &= a_{j-1}^i + p_j \cdot a_{m-1}^i, \text{ for } 1 \le j \le m-1. \end{aligned} \tag{2.12}$$

Given a general pentanomial of degree m,

$$F(t) = t^m + t^{s_1} + t^{s_2} + t^{s_3} + 1. \tag{2.13}$$

We can substitute (2.13) into (2.12) to have

$$\begin{aligned} a_0^{i+1} &= a_{m-1}^i \\ a_{s_1}^{i+1} &= a_{s_1-1}^i + a_{m-1}^i \\ a_{s_2}^{i+1} &= a_{s_2-1}^i + a_{m-1}^i \\ a_{s_3}^{i+1} &= a_{s_3-1}^i + a_{m-1}^i \\ a_j^{i+1} &= a_{j-1}^i, \text{ for } 1 \le j \le m-1 \text{ and } j \ne s_1, s_2, s_3. \end{aligned} \tag{2.14}$$

The same as the general trinomial of degree m given by

$$F(t) = t^m + t^s + 1, \tag{2.15}$$

where it can be substituted into (2.12) to have

$$\begin{aligned} a_0^{i+1} &= a_{m-1}^i \\ a_s^{i+1} &= a_{s-1}^i + a_{m-1}^i \\ a_j^{i+1} &= a_{j-1}^i, \text{ for } 1 \le j \le m-1 \text{ and } j \ne s. \end{aligned} \tag{2.16}$$

2.1.2 Conventional Systolic Structure

The conventional systolic multiplier based on NIST polynomials is shown in Fig. 2.1, where it consists of m PEs including three types of PEs: PE-1, PE-m, and regular PE (PE-2 through PE-$(m-1)$). The internal structures of these PEs are shown in Figs. 2.1(b), (c), and (d), respectively, where RC denotes the reduction cell (perform the operations of (2.14) and (2.16)). The internal structure of RC is shown in Fig. 2.2,

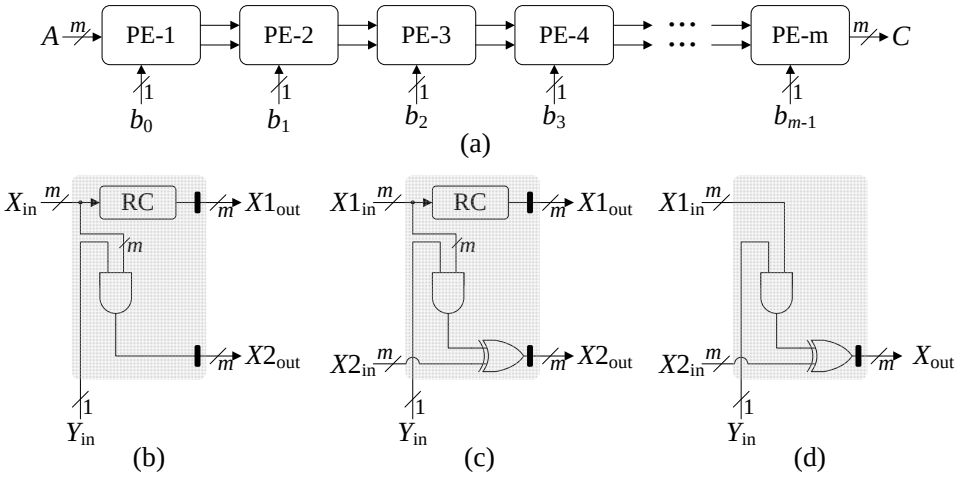

Figure 2.1: Conventional systolic structure for finite field multiplication over $GF(2^m)$ based on NIST pentanomials and trinomials. (a) Systolic structure. (b) Internal structure of PE-1. (c) Internal structure of a regular PE (PE-2 through PE-$(m-1)$). (d) Internal structure of PE-m.

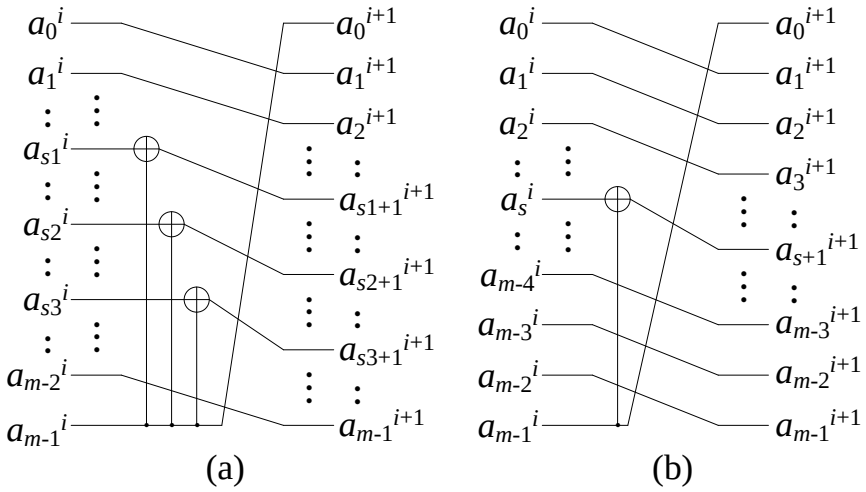

Figure 2.2: Detailed design of RC. (a) Detailed design of RC for pentanomial $F(t) = t^m + t^{s_1} + t^{s_2} + t^{s_3} + 1$. (b) Detailed design of RC for trinomial $F(t) = t^m + t^s + 1$.

where it involves three XORs for pentanomial and one XOR operation for trinomial. The latency of the structure in Fig. 2.1 is m cycles, where the duration of each cycle period is $T_A + T_X$ (T_A and T_X refer to the delays of an AND gate and an XOR gate, respectively).

2.1.3 Modified Low Register-Complexity Systolic Structure

For the structure of Figs. 2.1 and 2.2, we find that $(m-3)$ or $(m-1)$ registers in the RC of each PE pipeline unprocessed (without XOR operations) signals to the next PE. These registers can be removed if we propose another broadcasting strategy. As shown in Figs.

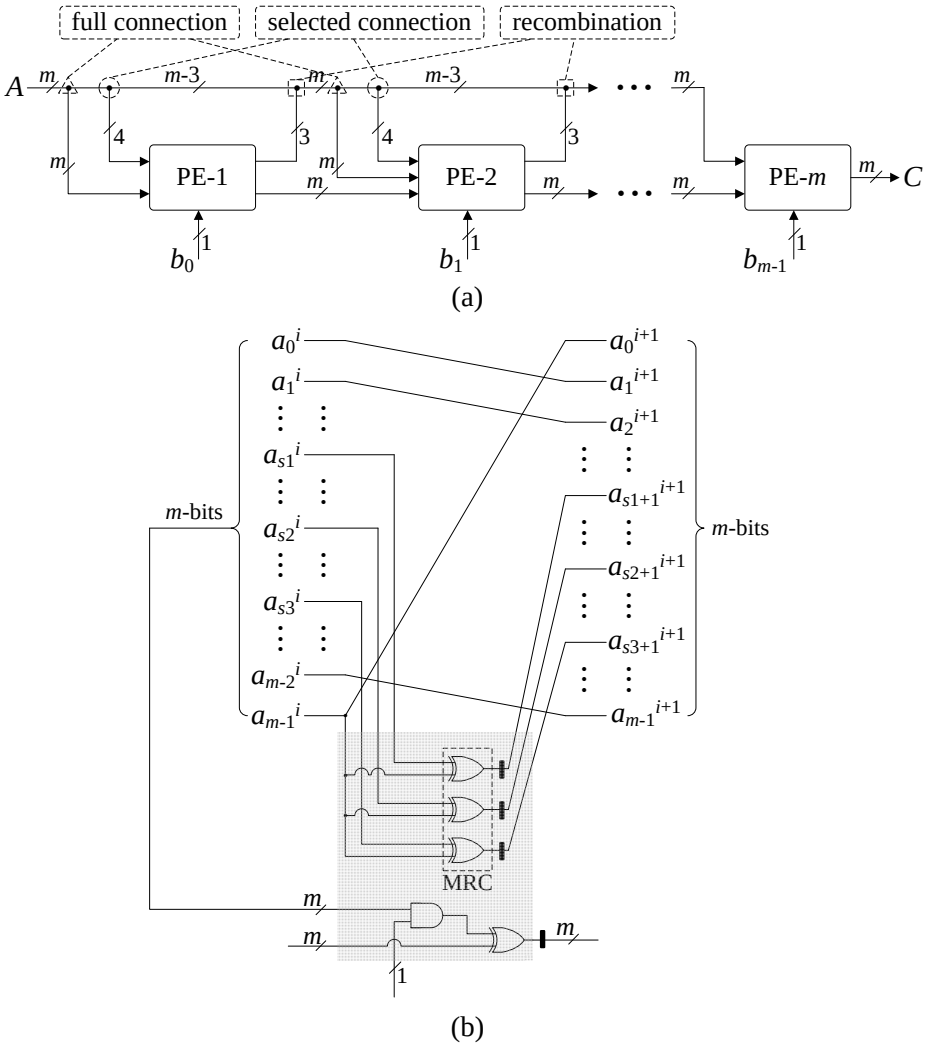

Figure 2.3: Detailed design of the modified systolic structure. (a) Modified systolic structure. (b) Modified reduction cell for pentanomial $F(t) = t^m + t^{s_1} + t^{s_2} + t^{s_3} + 1$.

2.3 and 2.4, a novel connection strategy including three types of connections is proposed, namely the full connection, selected connection, and recombination. For each PE, m bits of operand $A^{(i)}$ are directly fed to the AND gate in the PE for multiplication operation with one bit of operand B through full connection. Besides, 4 (or 2) bits of operand $A^{(i)}$ are selected to be fed into the modified RC (MRC) for XOR operations according to (2.14) and (2.16). The 3 (or 1) output bits from MRC will then be recombined into the operand $A^{(i+1)}$ to be used in next PE. Therefore, compared with the structure of Fig. 2.1, $(m-3)$ (or $(m-1)$) registers are reduced because of the proposed connection strategy. The modified structure has the same time-complexity as the previous one, but the register-complexity is significantly reduced.

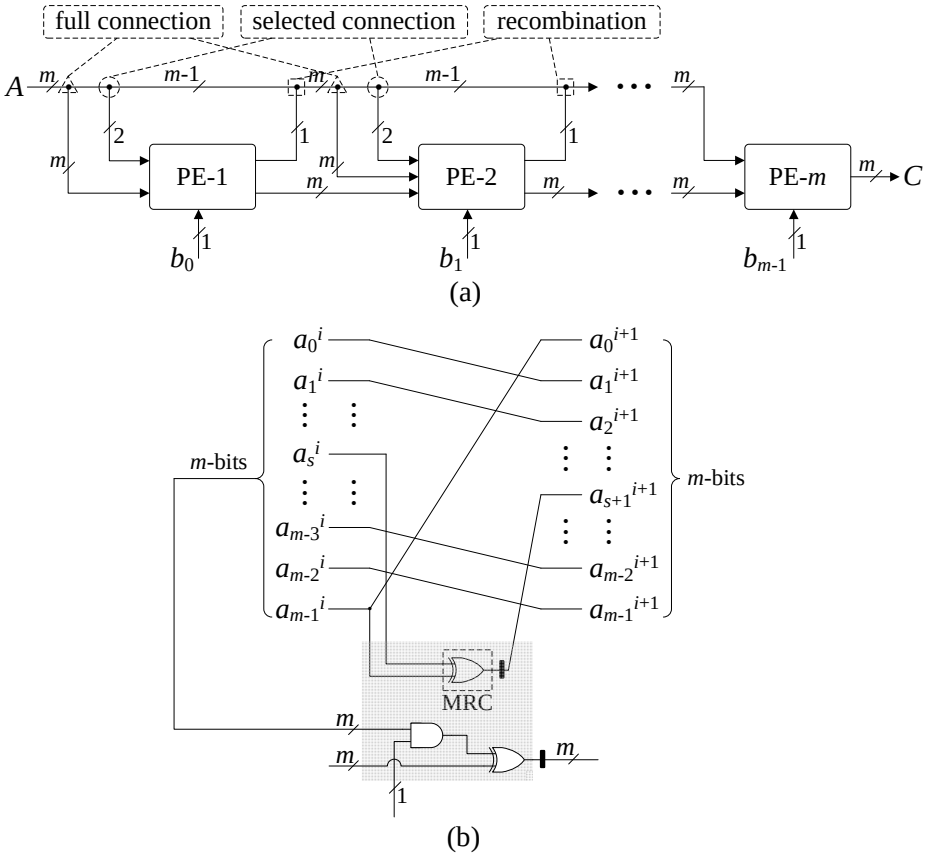

Figure 2.4: Detailed design of the modified systolic structure. (a) Modified systolic structure. (b) Modified reduction cell for trinomial $F(t) = t^m + t^s + 1$.

2.1.4 Area-Time Complexities

The area-time complexities of the proposed design in Figs. 2.3 and 2.4 are shown in Table 2.1, along with existing and conventional designs. It can be seen that the proposed design involves significantly less area-complexity when compared with competing ones, especially in terms of the register-complexity.

2.1.5 FPGA Implementation

We have also implemented these systolic structures in Table 2.1 to confirm the efficiency of proposed structures. We have synthesized these designs using Xilinx ISE 14.1 on Virtex 6 FPGA family with NIST pentanomial $F(t) = t^{163} + t^7 + t^6 + t^3 + 1$ and trinomial $F(t) = t^{233} + t^{74} + 1$. The results in terms of area-time-power complexities are shown in Table 2.2. It can be seen that the proposed structures outperform the existing ones, especially on register-complexity.

Table 2.1: COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS SYSTOLIC NIST POLYNOMIAL MULTIPLIERS

Design	AND	NAND	XOR	XNOR	Register	Critical-path delay	Latency
For NIST Pentanomials $F(t) = t^m + t^{s1} + t^{s2} + t^{s3} + 1$							
[11][1]	m^2	-	$m^2 + 2m +$ $2lm + 2l + 2$	-	$2m^2 - 2m -$ $2lm - 2l - 2$	$T_A + T_X$	$[m/(2l+2)+1$ $+\log_2(2l+2)]$
MBP-II [29][2]	m^2	-	$m^2 - 3m^2/d^2 +$ $5m/d + 3m^2/d -$ $m - 3 - s_1 + s_3$	-	$m^2 + md + m^2/d + 3m$ $-ds_1 + ds_3 + 5m/d -$ $s_1 + s_3 - 4d + 4m - 3$	$T_A + T_X$	$d + 1 + \log_2\lceil m/d \rceil$
Fig. 2.3[3]	m^2	-	$m^2 + 2m - 1$	-	$m^2 + 3m - 1$	$T_A + T_X$	m
For NIST Trinomials $F(t) = t^m + t^s + 1$							
[8]	-	m^2	$m^2 - 1$	-	$2m^2 - 2m$	$T_{NA} + T_X$	m
Fig. 2 [10]	-	m^2	$< 1.5m^2 + 0.5m + 1$	-	$1.5m^2 + 0.5m$	$2T_X$	$m + 1$
Fig. 3 [10]	-	m^2	$< 1.5m^2 + 0.5m + 1$	-	$1.5m^2 + 2m$	$T_{NA} + T_X$	$m + 2$
Fig. 8 [30]	-	m^2	-	$m^2 - 1$	$m^2 + 3m - 1$	$T_{NA} + T_{XN}$	$(m + 7)/2$
Fig. 2.4[3]	m^2	-	$m^2 - 1$	-	$m^2 + m$	$T_A + T_X$	m

T_{NA}: The delay time of an NAND gate. T_{XN}: The delay time of an XNOR gate.
[1]: Here l=min$\{m - s_1,\ s_1 - s_2,\ s_2 - s_3\}$. In [11], the authors have also used NAND and XNOR to replace part of original AND and XOR gates, we just list them as AND and XOR gates here, for simplicity of discussion.
[2]: The design of [29] is a low-latency design, where the original systolic array is decomposed into d arrays for parallel implementation.
[3]: For simplicity of discussion, we list here only the basic systolic structures. Although the proposed designs can be extended for low-latency implementation, which will be seen in Section III.

Table 2.2: FPGA IMPLEMENTATION RESULTS OF VARIOUS POLYNOMIAL-BASED MULTIPLIERS

Design	Area	Delay[1]	Power	ADP[2]	PDP[3]
For NIST Pentanomials $F(t) = t^{163} + t^7 + t^6 + t^3 + 1$					
[11]	52,482	1.695	1.613	88,957	2.734
MBP-II [30]	55,249	1.695	1.698	93,647	2.878
Fig. 2.3	28,149	1.695	0.865	47,713	1.466
For NIST Trinomials $F(t) = t^{233} + t^{74} + 1$					
[8]	111,840	1.695	3.435	189,569	5.822
Fig. 8 [31]	54,987	1.695	1.689	93,202	2.863
Fig. 2.4	54,522	1.695	1.675	92,415	2.839

Unit for area: number of slice register; Unit for delay: ns; Unit for power: W (Power is estimated at 100MHz).
[1]: Delay = Critical-Path.
[2]: ADP: Area-delay product = Area×Delay.
[3]: PDP: Power-delay product = Power×Delay.

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{s_1-1} \\ c_{s_1} \\ c_{s_1+1} \\ \vdots \\ \vdots \\ c_{s_2-1} \\ c_{s_2} \\ c_{s_2+1} \\ \vdots \\ \vdots \\ c_{s_3-1} \\ c_{s_3} \\ c_{s_3+1} \\ \vdots \\ \vdots \\ c_{m_1-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_{m_1-1} & a_{m_1-2} & a_{m_1-3} & \cdots \\ a_1 & a_0 & a_{m_1-1} & a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{s_1-1} & a_{s_1-2} & a_{s_1-3} & a_{s_1-4} & \cdots \\ a_{s_1} & a_{s_1-1}+a_{m_1-1} & a_{s_1-2}+a_{m_1-2} & a_{s_1-3}+a_{m_1-3} & \cdots \\ a_{s_1+1} & a_{s_1} & a_{s_1-1}+a_{m_1-1} & a_{s_1-2}+a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{s_2-1} & a_{s_2-2} & a_{s_2-3} & a_{s_2-4} & \cdots \\ a_{s_2} & a_{s_2-1}+a_{m_1-1} & a_{s_2-2}+a_{m_1-2} & a_{s_2-3}+a_{m_1-3} & \cdots \\ a_{s_2+1} & a_{s_2} & a_{s_2-1}+a_{m_1-1} & a_{s_2-2}+a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{s_3-1} & a_{s_3-2} & a_{s_3-3} & a_{s_3-4} & \cdots \\ a_{s_3} & a_{s_3-1}+a_{m_1-1} & a_{s_3-2}+a_{m_1-2} & a_{s_3-3}+a_{m_1-3} & \cdots \\ a_{s_3+1} & a_{s_3} & a_{s_3-1}+a_{m_1-1} & a_{s_3-2}+a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{m_1-1} & a_{m_1-2} & a_{m_1-3} & a_{m_1-4} & \cdots \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{s_1-1} \\ b_{s_1} \\ b_{s_1+1} \\ \vdots \\ b_{s_2-1} \\ b_{s_2} \\ b_{s_2+1} \\ \vdots \\ b_{s_3-1} \\ b_{s_3} \\ b_{s_3+1} \\ \vdots \\ b_{m_1-1} \end{bmatrix} \tag{2.17}$$

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{s-1} \\ c_s \\ c_{s+1} \\ \vdots \\ c_{m_2-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_{m_2-1} & a_{m_2-2} & a_{m_2-3} & \cdots \\ a_1 & a_0 & a_{m_2-1} & a_{m_2-2} & \cdots \\ a_2 & a_1 & a_0 & a_{m_2-1} & \cdots \\ a_3 & a_2 & a_1 & a_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{s-1} & a_{s-2} & a_{s-3} & a_{s-4} & \cdots \\ a_s & a_{s-1}+a_{m_2-1} & a_{s-2}+a_{m_2-2} & a_{s-3}+a_{m_2-3} & \cdots \\ a_{s+1} & a_s & a_{s-1}+a_{m_2-1} & a_{s-2}+a_{m_2-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{m_2-1} & a_{m_2-2} & a_{m_2-3} & a_{m_2-4} & \cdots \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{s-1} \\ b_s \\ b_{s+1} \\ \vdots \\ b_{m_2-1} \end{bmatrix} \tag{2.18}$$

2.2 Proposed Low Complexity Hybrid-Size Systolic Polynomial Multipliers

In this section, we first present the proposed hybrid-size polynomial multiplication algorithm, and then give the details of the proposed low complexity systolic structure based on detailed example.

2.2.1 Proposed Hybrid Polynomial Multiplication Algorithm

First of all, let us assume that the field orders of pentanomial and trinomial are m_1 and m_2, respectively. Without loss of generality, we can also assume $m_1 < m_2$ (though it is easy to extend to other cases like $m_1 > m_2$). Then, the pentanomial and trinomial multiplication process of (2.1)-(2.16) can be represented by the two matric-vector multiplications of (2.17) and (2.18), respectively. For simplicity of discussion, we can use $[C_P] = [A_P] \times [B_P]$ to represent (2.17), where $[C_P]$ and $[B_P]$ are bit-vectors contain all the bits of operands C and B, respectively, while $[A_P]$ is the multiplication process matrix of (2.17). Similarly, we can also use $[C_T] = [A_T] \times [B_T]$ to represent (2.18) (where $[C_T]$ and $[B_T]$ are bit-vectors of operands C and B, respectively, and $[A_T]$ is the multiplication process matrix of (2.18)).

Comparing $[A_P]$ of (2.17) with $[A_T]$ of (2.18), we can find that they all share with one matrix (m_1 by m_1) as

$$[A_C] = \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m_1-1} & a_{m_1-2} & a_{m_1-3} & \cdots & a_0 \end{bmatrix}, \tag{2.19}$$

where $[A_C]$ is a "half-cyclic" matrix, i.e., near half of the matrix are all "0", while the rest are all bits of "$a_0,\ a_1,\ a_2,\ ...,\ a_{m-1}$" (no bit-addition operations are involved). Substitute (2.19) into $[C_P] = [A_P] \times [B_P]$, we can have

$$[C_P] = ([A_{PX}] + [A_C]) \times [B_P], \tag{2.20}$$

where $[A_{PX}]$ is a matrix of m_1 by m_1 as

$$[A_{PX}] = \begin{bmatrix} 0 & a_{m_1-1} & a_{m_1-2} & \cdots \\ 0 & 0 & a_{m_1-1} & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & a_{m_1-1} & a_{m_1-2} & \cdots \\ 0 & 0 & a_{m_1-1} & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & a_{m_1-1} & a_{m_1-2} & \cdots \\ 0 & a_{m_1-1} & a_{m_1-2} + a_{m_1-1} & \cdots \\ 0 & 0 & a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 0 & a_{m_1-2} & a_{m_1-3} & \cdots \end{bmatrix}. \tag{2.21}$$

Similarly, one can also decompose $[C_T] = [A_T] \times [B_T]$ as

$$[C_T] = ([A_{TX}] + [A'_C]) \times [B_T], \tag{2.22}$$

where $[A'_C]$ is a m_2 by m_2 matrix mainly constituted with matrix $[A_C]$ and a number of

"0" to fill the rest empty positions, as:

$$[A'_C] = \begin{bmatrix} [A_C] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{2.23}$$

and the detail of matrix $[A_{TX}]$ is as follows:

$$[A_{TX}] = \begin{bmatrix} 0 & a_{m_2-1} & a_{m_2-2} & a_{m_2-3} & \cdots \\ 0 & 0 & a_{m_2-1} & a_{m_2-2} & \cdots \\ 0 & 0 & 0 & a_{m_2-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ 0 & a_{m_2-1} & a_{m_2-2} & a_{m_2-3} & \cdots \\ 0 & 0 & a_{m_2-1} & a_{m_2-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{m_1} & a_{m_1-1} & a_{m_1-2} & a_{m_1-3} & \cdots \\ a_{m_1+1} & a_{m_1} & a_{m_1-2} & a_{m_1-2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ a_{m_2-1} & a_{m_2-2} & a_{m_2-3} & a_{m_2-4} & \cdots \end{bmatrix}. \tag{2.24}$$

One can see that as (2.19) and (2.23) are all matrices with no XOR operations, (2.21) and (2.24) still involve certain amount of XOR operations. To facilitate the proposed systolic implementation, we can precalculate these XOR operations in advance, then the rest computation will be easier, e.g., the term of $a_{m_1-2} + a_{m_1-1}$ in the third column of matrix $[A_{PX}]$ can be computed in advance for (2.21) (similar rules apply to $[A_{TX}]$ of (2.24)). Such that (2.20) and (2.22) can be rewritten as

$$[C_P] = (\rho\{[A_{PX}]\} + [A_C]) \times [B_P], \tag{2.25}$$

and

$$[C_T] = (\rho\{[A_{TX}]\} + [A'_C]) \times [B_T], \tag{2.26}$$

where $\rho\{\cdot\}$ represents the above mentioned pre-XOR-computation operations.

Based on (2.17)-(2.26), we can have the proposed hybrid multiplication algorithm as given by Algorithm 2.1.

Algorithm 2.1 Proposed hybrid-size polynomial multiplication.

Inputs: A and B are the two field elements of polynomials in $GF(2^m)$ to be multiplied, S is the select signal.

Output: $C = A \cdot B \bmod F(t)$ ($F(t)$ can be pentanomial and trinomial).

1. Initialization step

1.1 $[A_P] = [A_C] + [A_{PX}]$.

1.2 $[A_T] = [A'_C] + [A_{TX}]$.

1.3 $[A_1] = 0$, $[A_2] = 0$ $[D] = 0$, $[D'] = 0$, $[E] = 0$, $[E_1] = 0$, $[E_2] = 0$.

2. Pre-XOR-computation step

2.1-a. $[A_1] = \rho\{[A_{PX}]\}$.

2.1-b. $[A_2] = \rho\{[A_{TX}]\}$.

3. Multiplication step

3.1. $[D] = [A_C] \times [B_P]$ (or $[D'] = [A'_C] \times [B_T]$).

3.2-a. $[E_1] = [A_1] \times [B_P]$.

3.2-b. $[E_2] = [A_2] \times [B_T]$.

4. Selection step

If $S = 0$, then

$[E] = [E_1] + [D]$.

Else,

$[E] = [E_2] + [D']$.

5. Final step

5.1. Operand $C \Leftarrow [C] = [E]$.

---

where for simplicity of discussion, $[E]$ is sized as m_1 by m_1 for pentanomial (the same size as $[D]$), and m_2 by m_2 for trinomial (the same size as $[D']$).

$$A_P = \underbrace{\begin{bmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 \\ a_3 & a_2 & a_1 & a_0 & \cdots & 0 \\ a_4 & a_3 & a_2 & a_1 & \cdots & 0 \\ a_5 & a_4 & a_3 & a_2 & \cdots & 0 \\ a_6 & a_5 & a_4 & a_3 & \cdots & 0 \\ a_7 & a_6 & a_5 & a_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{162} & a_{161} & a_{160} & a_{159} & \cdots & a_0 \end{bmatrix}}_{A_C} + \underbrace{\begin{bmatrix} 0 & a_{162} & a_{161} & a_{160} & \cdots & a_1 + a_{161} + a_{158} + a_{157} \\ 0 & 0 & a_{162} & a_{161} & \cdots & a_2 + a_{162} + a_{159} + a_{158} \\ 0 & 0 & 0 & a_{162} & \cdots & a_3 + a_{160} + a_{159} \\ 0 & a_{162} & a_{161} & a_{160} & \cdots & a_4 + a_{161} + a_{160} + a_1 + a_{161} + a_{158} + a_{157} \\ 0 & 0 & a_{162} & a_{161} & \cdots & a_5 + a_{162} + a_{161} + a_2 + a_{162} + a_{159} + a_{158} \\ 0 & 0 & 0 & a_{162} & \cdots & a_6 + a_{162} + a_3 + a_{160} + a_{159} \\ 0 & a_{162} & a_{161} & a_{160} & \cdots & a_7 + a_4 + a_{161} + a_{160} + a_1 + a_{161} + a_{158} + a_{157} \\ 0 & a_{162} & a_{162} + a_{161} & a_{161} + a_{160} & \cdots & a_8 + a_5 + a_{162} + a_{161} + a_2 + a_{162} + a_{159} + a_{158} + a_1 + a_{161} + a_{158} + a_{157} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{160} + a_{157} + a_{156} \end{bmatrix}}_{A_{PX}}$$

(*a*)

$$A_T = \underbrace{\begin{bmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 \\ a_3 & a_2 & a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{73} & a_{72} & a_{71} & a_{70} & \cdots & 0 \\ a_{74} & a_{73} & a_{72} & a_{71} & \cdots & 0 \\ a_{75} & a_{74} & a_{73} & a_{72} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{162} & a_{161} & a_{160} & a_{159} & \cdots & a_0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{A_C'} + \underbrace{\begin{bmatrix} 0 & a_{232} & a_{231} & a_{230} & a_{229} & a_{228} & \cdots & a_1 + a_{160} \\ 0 & 0 & a_{232} & a_{231} & a_{230} & a_{229} & \cdots & a_2 + a_{161} \\ 0 & 0 & 0 & a_{232} & a_{231} & a_{230} & \cdots & a_3 + a_{162} \\ 0 & 0 & 0 & 0 & a_{232} & a_{231} & \cdots & a_4 + a_{163} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{74} \\ 0 & a_{232} & a_{231} & a_{230} & a_{229} & a_{228} & \cdots & a_{75} + a_1 + a_{160} \\ 0 & 0 & a_{232} & a_{231} & a_{230} & a_{229} & \cdots & a_{76} + a_2 + a_{161} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_{163} + a_{89} \\ a_{163} & a_{162} & a_{161} & a_{160} & a_{159} & a_{158} & \cdots & a_{164} + a_{90} \\ a_{164} & a_{163} & a_{162} & a_{161} & a_{160} & a_{159} & \cdots & a_{165} + a_{91} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{232} & a_{231} & a_{230} & a_{229} & a_{228} & a_{227} & \cdots & a_0 + a_{159} \end{bmatrix}}_{A_{TX}}$$

(*b*)

Figure 2.5: Detailed example according to Algorithm 2.1. (a) Pentanomial example of $GF(2^{163})$. (b) Trinomial example of $GF(2^{233})$.

2.2.2 Detailed Example and Computation Steps

To have a clear understanding of the proposed algorithm and design strategy, we give here an example of defining pentanomial $F(t) = t^{163} + t^7 + t^6 + t^3 + 1$ and trinomial $F(t) = t^{233} + t^{74} + 1$. Then, according to Algorithm 2.1, we have the equations as shown in Fig. 2.5.

Then, we can re-express $[A_C]$ of Fig. 2.5 as $[A_C] = [A_C^{(0)}\ A_C^{(1)}\ A_C^{(2)}\ \cdots\ A_C^{(162)}]$, where $A_C^{(0)}$ through $A_C^{(162)}$ represents every column of $[A_C]$, respectively. Similarly, we can have

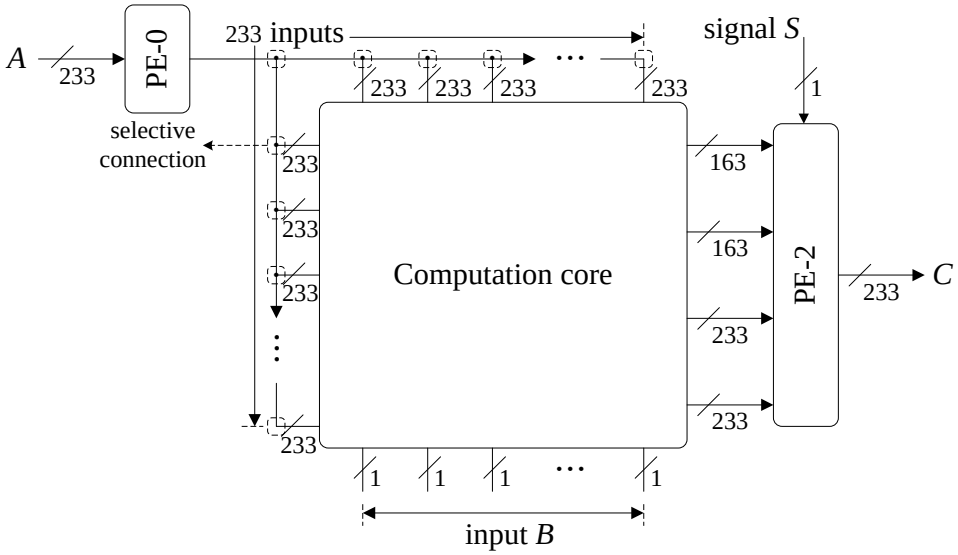

Figure 2.6: Proposed hybrid-size systolic multiplier, where the dotted box denotes selective connection.

$[A'_C] = [A'^{(0)}_C \ A'^{(1)}_C \ A^{(2)}_C \ \cdots \ A'^{(232)}_C]$ where $A'^{(0)}_C$ through $A'^{(232)}_C$ are the corresponding columns of $[A'_C]$.

Likewise, after the pre-XOR-computation operations, we can have $\rho\{[A_{PX}]\} = [A^{(0)}_{PX} \ A^{(1)}_{PX} \ A^{(2)}_{PX} \ \cdots \ A^{(162)}_{PX}]$ and $\rho\{[A_{TX}]\} = [A^{(0)}_{TX} \ A^{(1)}_{TX} \ A^{(2)}_{TX} \ \cdots \ A^{(232)}_{TX}]$, where $A^{(0)}_{PX}$ through $A^{(162)}_{PX}$ and $A^{(0)}_{TX}$ through $A^{(232)}_{TX}$ are every columns of $\rho\{[A_{PX}]\}$ and $\rho\{[A_{TX}]\}$, respectively.

Then, the rest multiplication step according to Algorithm 2.1 is:

$$\begin{aligned}[E] = {} & A^{(0)}_C b_0 + A^{(1)}_C b_1 + \cdots + A^{(162)}_C b_{162} \\ & + A^{(0)}_{PX} b_0 + A^{(1)}_{PX} b_1 + \cdots + A^{(162)}_{PX} b_{162},\end{aligned} \tag{2.27}$$

or

$$\begin{aligned}[E] = {} & A'^{(0)}_C b_0 + A'^{(1)}_C b_1 + \cdots + A'^{(232)}_C b_{232} \\ & + A^{(0)}_{TX} b_0 + A^{(1)}_{TX} b_1 + \cdots + A^{(232)}_{TX} b_{232}.\end{aligned} \tag{2.28}$$

Then, we can have the following structure.

2.2.3 Proposed Hybrid-Size Systolic Structure

The overall structure of hybrid-size systolic multiplier based on Algorithm 2.1 (where the example of Fig. 2.5 is employed) is shown in Fig. 2.6, which consists of one computation core and two extra PEs. PE-0 performs the pre-XOR-computation of operand A according

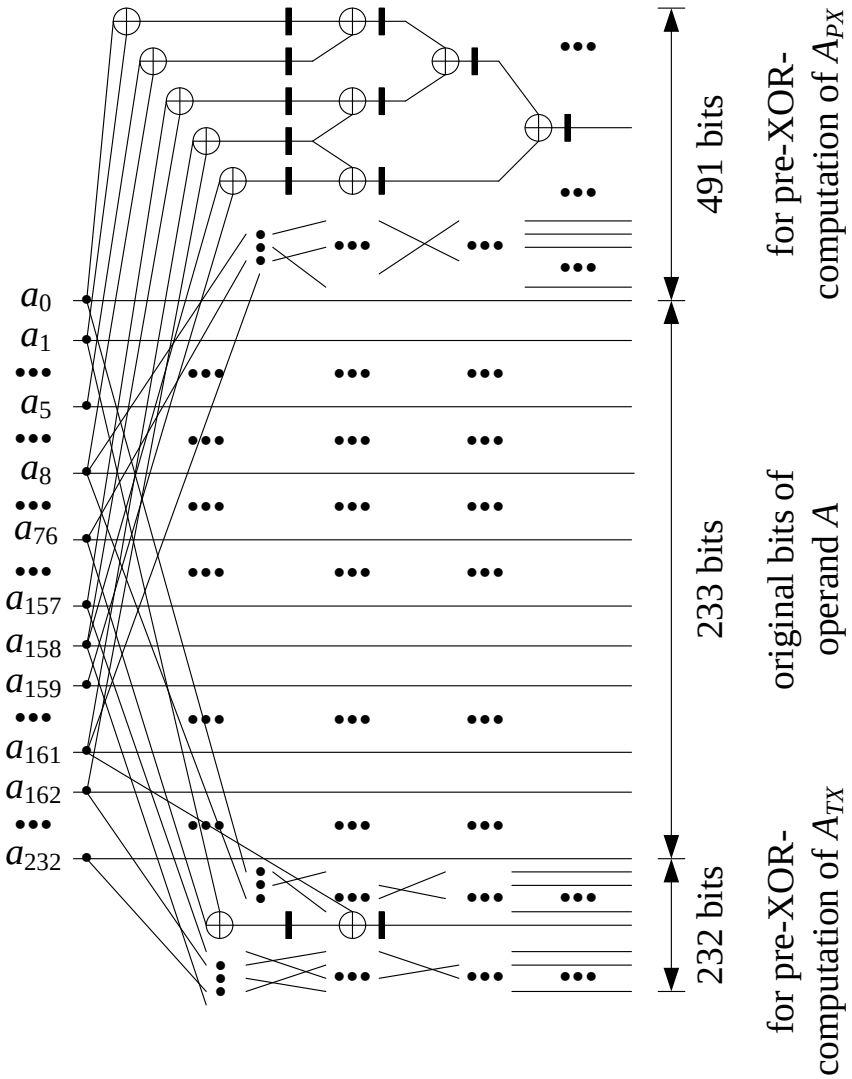

Figure 2.7: Detailed internal structure of PE-0.

to Step 2 of Algorithm 2.1. The detailed structure of PE-0 is shown in Fig. 2.7, where we use multi-stage pipelining technique to maintain the critical-path of PE-0 as T_X (T_X is the delay time of an XOR gate, and according to Fig. 2.5, the maximum time for pre-XOR-computation operations is $4T_X$ for $[A_{PX}]$ and $2T_X$ for $[A_{TX}]$, respectively). Note that for those bits without XOR operations, we do not need to add register for pipelining. The output bits of PE-0 (in total 956 bits: 233 bits of a_0 through a_{232}, 491 bits from pre-XOR-computation of $[A_{PX}]$, and 232 bits from pre-XOR-computation of $[A_{TX}]$) are selected to be connected to the computation core (as indicated by the dotted box), where the computation core mainly executes the Steps of 3 and 4 of Algorithm 2.1.

The internal structure of the computation core is shown in Fig. 2.8, where four arrays of PEs are involved. The first (from top) array is for the computation of $[A_C]$, the

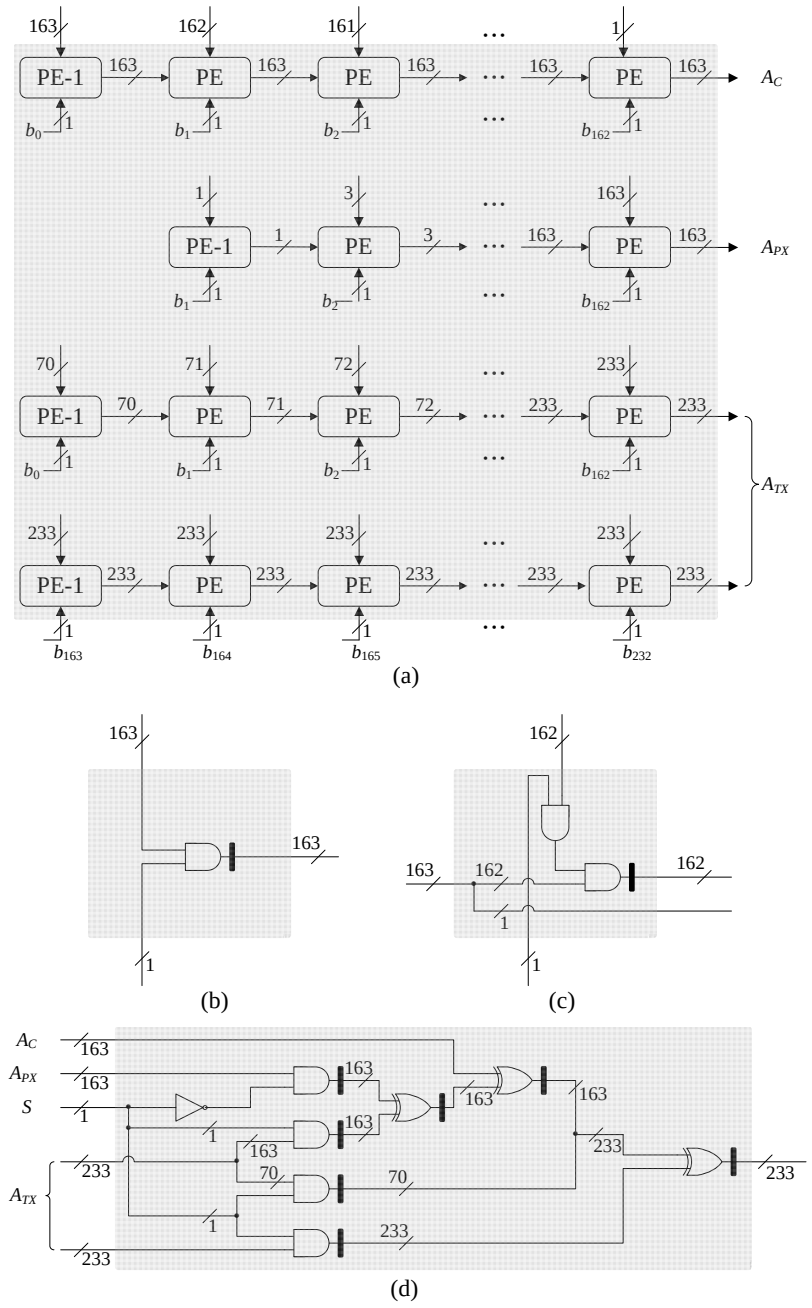

Figure 2.8: Detailed structure of the proposed multiplier, where the black boxes represent the registers. (a) Internal structure of the computation core. (b) Example of detailed design of PE-1. (c) Example of detailed design of a regular PE. (d) Detailed design of PE-2.

second array is for $[A_{PX}]$, and the third and fourth arrays are for the calculation of $[A_{TX}]$ (decomposed into two arrays for parallel processing). The computation core contains 558 PEs (where two kinds of PEs are being used), and Figs. 2.8(b) and (c) gives the example of the detailed design of PE-1 and regular PE, respectively. PE-1 performs multiplication between one selected operand and one bit of operand B, and then produces the result to its right. The regular PE performs the multiplication between one selected operand and

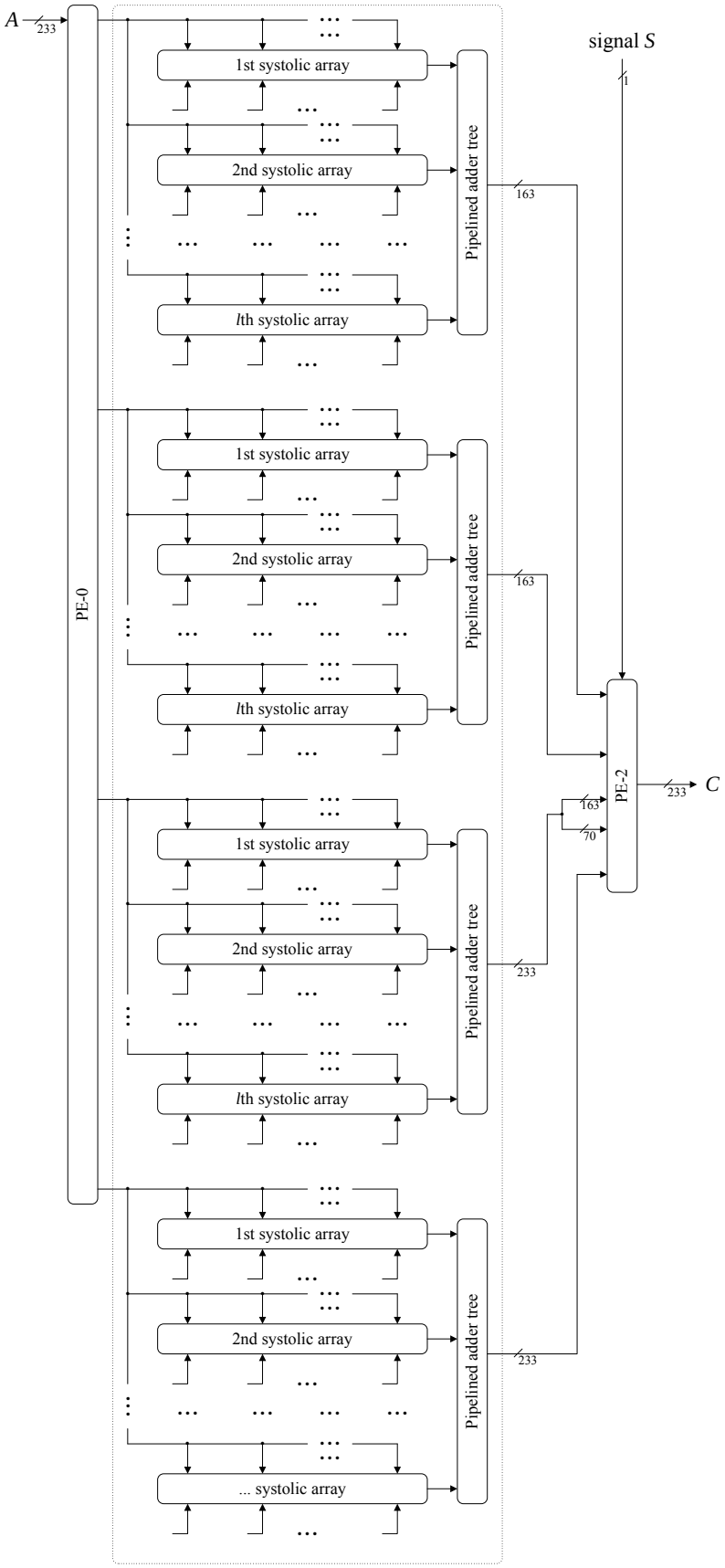

Figure 2.9: Proposed low-latency systolic structure.

Table 2.3: COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS SYSTOLIC POLYNOMIAL MULTIPLIERS

Design	AND	NAND	XOR	XNOR	Register	Latency	Critical-path Delay
For NIST Pentanomial $F(t) = t^{163} + t^7 + t^6 + t^3 + 1$							
Fig. 5(a)	$27,658$	-	$27,986$	-	$28,312$	163	$T_A + T_X$
[11]	$26,569$	-	$27,225$	-	$52,482$	44	$T_A + T_X$
MBP-II [29]	$26,569$	-	$26,890$	-	$53,136$	164	$T_A + T_X$
For NIST Trinomial $F(t) = t^{233} + t^{74} + 1$							
Fig. 5(b)	$58,100$	-	$58,169$	-	$58,565$	233	$T_A + T_X$
[8][1*]	$54,289$	-	$55,454$	-	$121,684$	32	$T_A + T_X$
Fig. 3* [10]	$54,289$	-	$81,551$	-	$81,900$	235	$T_A + T_X$
For Hybrid-size Polynomial $m_1 = 163$, $m_2 = 233$							
Fig. 2.6	$72,858$	-	$72,882$	-	$74,940$	233	$T_A + T_X$

T_A and T_X are the critical-path delay of AND gate and XOR gate, respectively.
*: The authors have used NAND and XNOR, we just list them as AND and XOR gates here for simplicity discussion.
[1]: The super-systolic structure.
[2]: The structure with $e = 2$ and $d = 1$ (PEs from MS-I).

one bit of operand B first, and then adds the input from the left PE to yield the final result to the right.

The detailed design of PE-2 of Fig. 2.6 is also shown in Fig. 2.8(d), where the selection signal S works to produce the output C according to the field-size selected (as Step 4 of Algorithm 2.1). Note that for pentanomial, only the first 163 output bits (from top) will be counted as output of C. The proposed systolic structure produces a result in 170 cycles (PE-0 is 4 cycles, computation core is 163 cycles, and PE-2 is 3 cycles), and the critical-path of each cycle is $T_A + T_X$ (T_A denotes the delay time of an AND gate).

2.2.4 Low-Latency Structure

For practical implementations, we can further reduce the latency of proposed structure in Figs. 2.6 and 2.8. Let $233 = l \cdot d + x$, where $0 \leq x \leq d$ (it is applicable to any field-size of m). Assuming $x = 0$ for simplicity, however, it can be extended to $x \neq 0$. Then, we can decompose each array in the computation core of Fig. 2.8(a) divided into l arrays as shown in Fig. 2.9 (each array will have d PEs).

Table 2.4: FPGA IMPLEMENTATION RESULTS OF VARIOUS POLYNOMIAL-BASED MULTIPLIERS

Design	Area	Delay[1]	Power	ADP[2]	PDP[3]
Fig. 2.3	28,149	1.695	0.865	47,713	1.446
Fig. 2.4	54,552	1.695	1.675	92,415	2.839
Proposed	77,006	1.695	2.364	130,525	4.007

Unit for area: number of slice register; Unit for delay: *ns*; Unit for power: W (Power is estimated at 100MHz).
[1]: Delay = Critical-Path.
[2]: ADP: Area-delay product = Area×Delay.
[3]: PDP: Power-delay product = Power×Delay.

2.3 Area and Time Complexity

2.3.1 Theoretical Comparison

The area and time complexities in terms of logic gate count, register count, latency, and critical-path delay of the proposed and existing structures are listed in table 2.3. The comparison contains two parts: pentanomials and trinomials. According to the table, the register-complexity of the multiplier in [11] is 52,482, and MBP-II in [29] has 53,136 registers, while our modified structure of pentanomial as shown in Fig. 2.5(a) requires the register-complexity of 28,312, which is 46.05% less than [11], and 46.72% less than [29]. Similar condition is found in comparison of trinomials. Our modified structure of trinomial as shown in Fig. 2.5(b) has lower register-complexity of 58,565 compared with the existing ones, among which the structure of Fig. 3 in [10] has 81,900 registers, while the super-systolic structure in [8] need even more registers, which is 121,684. Our modified trinomial structure saves 28.49% and 51.87% registers when compared to [10] and [8], respectively. Our proposed hybrid-size systolic polynomial multiplier as shown in Fig. 2.6 has much less area complexities than any combination of the listed pentanomials and trinomials.

2.3.2 FPGA Implementation

We also implement our proposed design on hardware (FPGA platform) to investigate the performance of the proposed low-complexity, low-latency hybrid-size systolic polynomial

multiplier as shown in Fig. 2.9. We have synthesized the proposed structure using Xilinx ISE 14.1 on Virtex 6 FPGA family. The results in terms of area-time-power complexities are shown in Table 2.4. It can be seen that our proposed low-complexities hybrid-size systolic multiplier outperform the combination of the existing ones.

Chapter 3

Low Critical-Path Low-Complexity Digit-Level Systolic Gaussian Normal Basis Multiplier

In this chapter, we propose a digit-level systolic Gaussian normal basis multiplier which can achieve low critical-path and low register-complexity.

3.1 Review of the Existing DL Systolic GNB Multiplier

Normal basis $N = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$ exists in the finite field $GF(2^m)$ over $GF(2)$ for any positive integer m, where β is called normal element. Each field element in $GF(2^m)$, take $A = (a_0, a_1, \cdots, a_{m-1})$ as an example, can be represented as a linear combination of the elements in N, i.e., $A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = a_0\beta + a_1\beta^2 + \cdots + a_{m-1}\beta^{2^{m-1}}$, where $a_i \in GF(2^m)$, $0 \leq i \leq m-1$. Assume that $m > 1$ and $T > 1$ are two integers. Let $p = mT + 1$ be a prime number and $\gcd(mT/k, m) = 1$, where k is the multiplication order of 2 modulo p. Then, the normal basis $N = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$ of $GF(2^m)$ over $GF(2)$ is called the Gaussian normal basis of type T [45].

The multiplication over GNB can be performed based on a multiplication matrix $\mathbf{M}_{m\times m}$ [49]. Let A and B be two field elements over $GF(2^m)$,

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}, \; B = \sum_{j=0}^{m-1} b_j \beta^{2^j}. \tag{3.1}$$

then we can have their product C as

$$C = (c_0, c_1, \cdots, c_{m-1}) = AB = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i + 2^j}. \tag{3.2}$$

Let us define $\mu_{i,j} = \beta^{2^i+2^j} \in GF(2^m)$ as a field element, where $0 \leq i, j \leq m-1$. Then, with respect to N, one can have

$$\mu_{i,j} = \sum_{l=0}^{m-1} \mu_{i,j}^{(l)} \beta^{2^l}, \tag{3.3}$$

substituting (3.3) into (3.2), one can have

$$\begin{aligned} C &= \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \sum_{l=0}^{m-1} \mu_{i,j}^{(l)} \beta^{2^l} \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{l=0}^{m-1} a_i b_j \mu_{i,j}^{(l)} \beta^{2^l}. \end{aligned} \tag{3.4}$$

Let us define the l-th coordinate of C as

$$c_l = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \mu_{i,j}^{(l)}, \tag{3.5}$$

which can also be represented in a matrix form as

$$c_l = \underline{a} \cdot \mathbf{M}^{(l)} \cdot \underline{b}^{tr}, \quad 0 \leq l \leq m-1, \tag{3.6}$$

where $\underline{a} = [\, a_0, a_1, \cdots, a_{m-1} \,]$ denotes the row vector corresponding to the field element A, and $\underline{b}^{tr}$ represents the matrix transpose of row vector $\underline{b} = [\, b_0, b_1, \cdots, b_{m-1} \,]$ which corresponds to the field element B. $\mathbf{M}^{(l)}$ and can be obtained from the l-fold right and down cyclic shifting of the multiplication matrix $\mathbf{M} = \mathbf{M}^{(0)}$. Then, we can write the product C as

$$C = \sum_{l=0}^{m-1} c_l \beta^{2^l}. \tag{3.7}$$

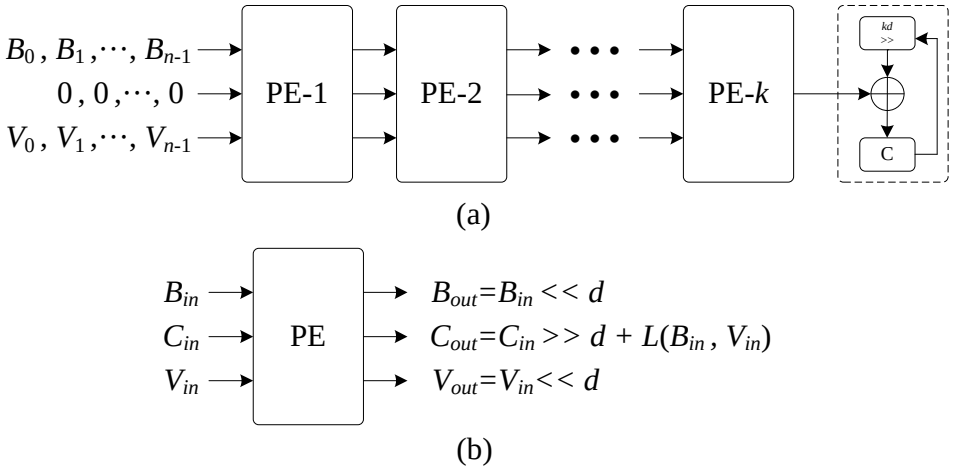

Figure 3.1: (a) Existing DL systolic GNB multiplier over $GF(2^m)$ [48]. (b) Detailed structure of PE.

Algorithm 3.1 Existing DL Systolic multiplication [48].

Inputs: $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ are two field elements over $GF(2^m)$, where m is an odd number.

Output: $C = (c_0, c_1, \cdots, c_{m-1}) = A \cdot B$.

1. $C = 0$
2. $V = \overline{A} \gg 1$, where $\overline{A} = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0)$.
3. for $i = 0$ to $n - 1$ do.
4. $\quad C_i = 0$.
5. $\quad V_{i,0} = V \gg kid + (k-1)d$.
6. $\quad B_{i,0} = B \gg kid + (k-1)d$.
7. $\quad$ for $j = 1$ to k do.
8. $\quad\quad C_i = (C_i \gg d) + L(V_{i,j-1}, B_{i,j-1})$.
9. $\quad\quad V_{i,j} = V_{i,j-1} \ll d$ and $B_{i,j} = B_{i,j-1} \ll d$.
10. $\quad$ end for.
11. $\quad C = (C \gg kd) + C_i$.
12. end for.

Recently, a low-latency systolic GNB multiplier has been proposed in [48]. Algorithm 3.1 describes the existing systolic GNB multiplication. Let us define $q = \lceil m/d \rceil$, where d is the digit size and $1 \leq d \leq m$. Then, the product $C = AB$ can be performed as $C = \sum_{i=0}^{q-1} L^{2^{id}}(V \gg id,\ B \gg id)$, where $L(V, B) = \sum_{j=0}^{d-1} J^{2^j}(V \gg j,\ B \gg j)$. Let us define n and k as two integers that satisfy $q = kn$, then, we can get the partial product C_i by $C_i = \sum_{j=0}^{k-1} L^{2^{id}}(V_i \gg jd,\ B_i \gg jd)$. Thus, one can decompose the product C into n-term partial products, which is $C = C_0 + C_1^{2^{kd}} + \cdots + C_{n-1}^{2^{(n-1)kd}} = (((C_{n-1})^{2^{kd}} + C_{n-2})^{2^{kd}} + \cdots)^{2^{kd}} + C_0$. Each partial product can be written as $C_i = (((L(\overline{V_i},\ \overline{B_i}))^{2^d} + L(\overline{V_i} \ll d,\ \overline{B_i} \ll d))^{2^d} + \cdots +)^{2^d} + L(\overline{V_i} \ll (k-1)d,\ \overline{B_i} \ll (k-1)d)$.
Based on Algorithm 3.1, Fig. 3.1(a) depicts the existing DL systolic GNB multiplier over $GF(2^m)$ [48]. We can see that the existing multiplier contains k processing elements (PEs) and one accumulating modular (AM). Each PE is carried out by the Steps 8 and 9 of Algorithm 3.1, and the AM is computed by Step 11.
Fig. 3.1(b) presents the detailed internal structure of PEs. According to Algorithm 3.1, we need to define the output B of PE_j as $B_{i,j}$ to compute the partial product C_i. Two elements V_i and B_i, $1 \leq i \leq n-1$, are fed to the multiplier from left cyclically to compute the partial product C_i recursively. The latency of the multiplier is $(k + n)$ cycles, i.e., it takes $(k + n)$ cycles to get the final product $C = AB$. The critical-path delay of the existing systolic GNB multiplier is $T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d + 1) \rceil)T_X$, where T_A and T_X are the delay of an AND gate and an XOR gate, respectively. The structure of Fig. 3.1 has $\lceil \sqrt{\frac{m}{d}} \rceil dm$ AND gates, $\{\leq \lceil \sqrt{\frac{m}{d}} \rceil d(m-1)(T-1)/2 + (1 + \lceil \sqrt{\frac{m}{d}} \rceil d)m\}$ XOR gates, and $(1 + 3\lceil \sqrt{\frac{m}{d}} \rceil)m$ bit-registers.

3.2 Proposed Algorithm

In this section, we present our proposed algorithms separately to reduce the critical-path and register-complexity.

3.2.1 Low Critical-Path Delay

From the structure of Fig. 3.1(b) and Algorithm 3.1, we find that inside the PE, the operation $L(V_{in}, B_{in})$ consists of three main parts, i.e., two additions and one multiplication. The first addition consists of a recombined addition operation (RAO), which performs

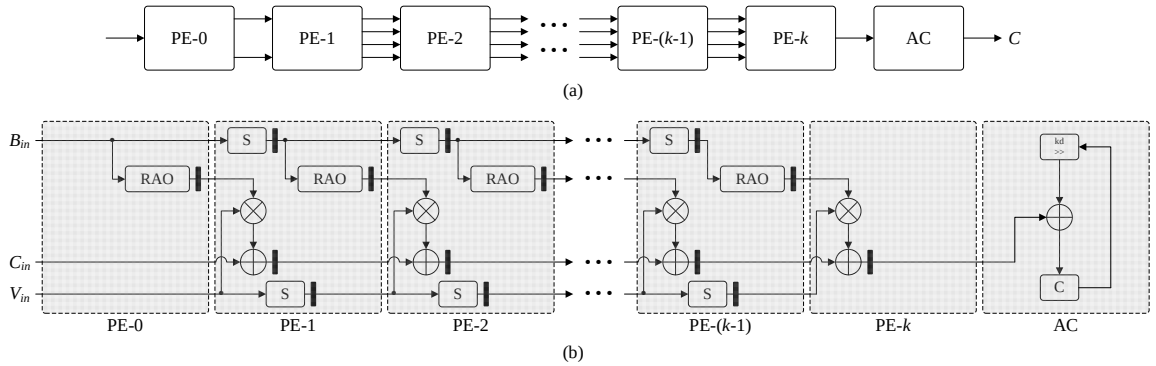

Figure 3.2: Proposed strategy of designing low critical-path delay DL systolic GNB multiplier over $GF(2^m)$, where S box refers to shift operation. (a) Proposed DL systolic GNB multiplier. (b) Detailed structure of PEs, where black boxes denote registers.

the addition of a series of reconstructed operand B itself. All the bits of operand B are positionally-recombined and some of them are added together to form the new bits according to Algorithm 3.1. Then, the multiplication is performed between the operand B after the recombined addition and the operand V. The result of multiplication is added with the input from the previous PE and then produces the result to the next PE on its right. The critical-path delay of this structure is thus $T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d+1) \rceil) T_X$. Although the structure in Fig. 3.1 is efficient in implementation, it can still be improved further, e.g., the critical-path delay needs to be shortened for practical high-performance applications. Here, we introduce a novel algorithm in which the critical-path delay is shortened by performing the RAO of the operand B in advance.

Fig. 3.2 depicts the proposed strategy of achieving low critical-path delay implementation based on the existing DL systolic multiplier over $GF(2^m)$. Form Fig. 3.2(a), one can see that the structure consists of $(k+1)$ PEs and one accumulation cell (AC). The detailed internal operational-structure of these components are presented in Fig. 3.2(b). First of all, an additional PE (PE-0) is added to perform the first RAO of operand B before PE-1. The result of RAO from PE-0 is then yielded to PE-1 to perform the multiplication and then the addition. Meanwhile, we still have the RAO in PE-1 which yields its result to the next PE on its right. "S box" performs the shifting of the bits of both operands V and B. After $(k+1)$ clock cycles, we get the first partial product. All the partial products C_is are recursively fed into the AC to get the final result C after n clock cycles. The critical-path delay of this proposed architecture is thus $T_A + (\lceil \log_2(d+1) \rceil) T_X$ (for NIST recommended GNB, $(\lceil \log_2(d+1) \rceil) T_X$ is larger than $\lceil \log_2 T \rceil$).

According to the strategy shown in Fig. 3.2, we derive here the modified low critical-path delay DL systolic multiplication algorithm as presented in the proposed Algorithm 3.2.

Algorithm 3.2 Low critical-path delay DL Systolic Multiplication.

Inputs: $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ are two field elements over $GF(2^m)$, where m is an odd number.

Output: $C = (c_0, c_1, \cdots, c_{m-1}) = A \cdot B$.

1. $C = 0$
2. $V = \overline{A} \gg 1$, where $\overline{A} = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0)$.
3. for $i = 0$ to $n - 1$ do.
4. $C_i = 0$.
5. $V_{i,0} = V \gg kid + (k-1)d$.
6. $B_{i,0} = B \gg kid + (k-1)d$.
7. $B^*_{i,0} = RAO(B_{i,0})$.
8. for $j = 1$ to k do.
9. $C_i = (C_i \gg d) + MA(V_{i,j-1}, B^*_{i,j-1})$.
10. $V_{i,j} = V_{i,j-1} \ll d$ and $B_{i,j} = B_{i,j-1} \ll d$.
11. $B^*_{i,j} = RAO(B_{i,j})$.
12. end for.
13. $C = (C \gg kd) + C_i$.
14. end for.

According to Algorithm 3.2, we perform the first RAO of the operand B independently in PE-0, which corresponds to Step 7. Each PE (from PE-1 to PE-k) is carried out by the computation of the Steps 9, 10, and 11 of Algorithm 3.2, where MA denotes multiplication and addition inside each PE, and AC is carried out in Step 13. By computing the RAO of operand B in advance in each PE, we have shortened the critical-path delay of the structure of Fig. 3.1.

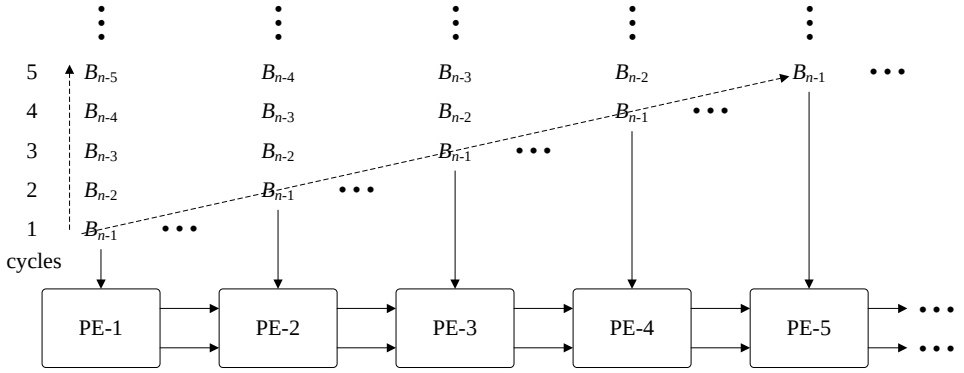

Figure 3.3: Data pipelining of operand B among PEs for the structure of Fig. 3.1, where the diagonal line represents data flow between PEs, and the vertical line represents data flow in one PE.

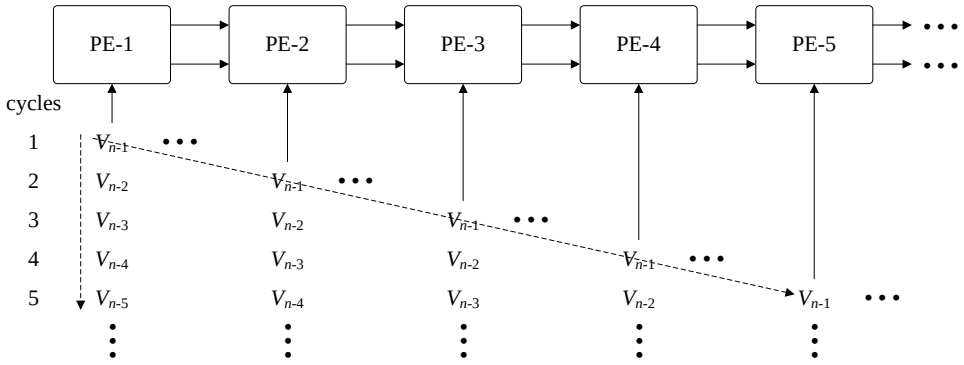

Figure 3.4: Data pipelining of operand V among PEs for the structure of Fig. 3.1, where the diagonal line represents data flow between PEs, and the vertical line represents data flow in one PE.

3.2.2 Low Register-Complexity

Systolic structure sometimes suffers from large register-complexity, as all the PEs in the array are uniform and fully-pipelined (there are a lot of registers in the PEs). In this subsection, we propose a novel strategy to reduce the corresponding registers among PEs. As seen from Fig. 3.1(b), there are generally two types of registers equipped for one PE: Type-one for operand pipelining (after bits shifting, the top one for operand B and the bottom one for operand V); another one for pipelining of computation (the registers used to pipeline the data after the $L(B_{in}, V_{in})$ operation). The registers used to pipeline the computational data are critical to the correctness of final output while the registers for pipelining the shifted operands (the top and bottom ones) are relatively less important. Based on the above consideration, we propose here a novel strategy to reduce the registers related to the pipelining of the shifted operands (the top and bottom ones). Let us first consider the data pipelining of shifted-operand B among PEs in the existing design of

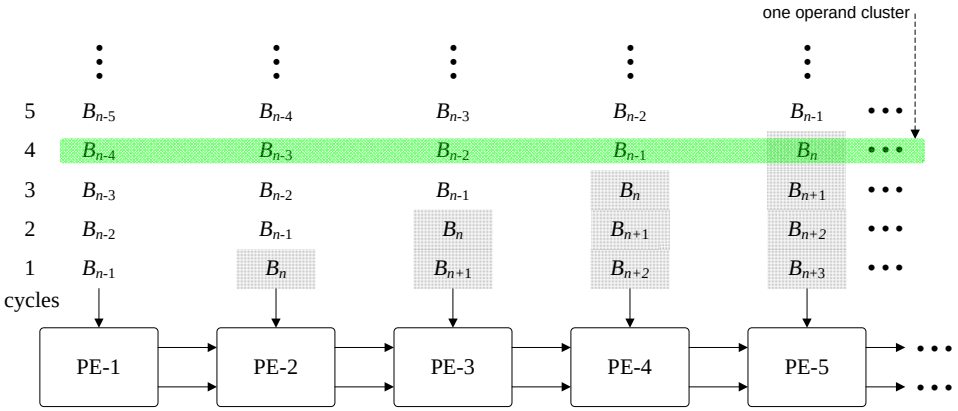

Figure 3.5: Data pipelining of operand B among PEs with added operands for the structure of Fig. 3.1, where the gray area represents all added operands, and the green area represents one specific operand cluster fed to all PEs.

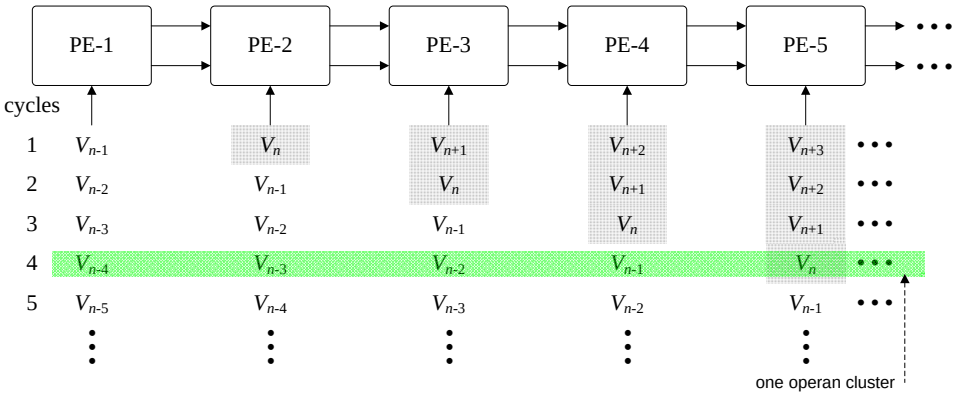

Figure 3.6: Data pipelining of operand V among PEs with added operands for the structure of Fig. 3.1, where the gray area represents all added operands, and the green area represents one specific operand cluster fed to all PEs.

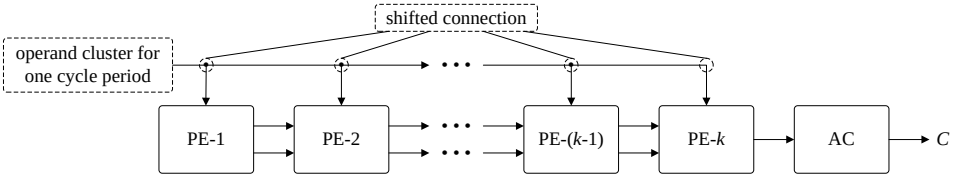

Figure 3.7: Example of rearranging data pipelining by using operand cluster B'_i.

[48]. It is seen that, in Fig. 3.3, the shifted-operand's subscript (the subscript denotes the degree of shifting, according to Fig. 3.1) increases one per cycle for a single PE (for neighboring PEs, within the same cycle, the subscript increases with the numbering of PE). The pipelining of the bits of shifted-operand V is similar to the shifted-operand B, as shown in Fig. 3.4.

To give the detailed register-reduction strategy, we can first add extra operands with

corresponding subscript to fill the data flow table (highlighted gray areas), such that the operand's subscript increases one per cycle for a single PE and one for neighboring PEs, as shown in Figs. 3.5 and 3.6. For simplicity of discussion, we can define all the shifted-operands for one specific clock cycle as one operand cluster, as shown in Figs. 3.5, 3.6, and 3.7. Thus, after the required clock cycles, we can still get the same output as that in Fig. 3.1 since the corresponding operand will still be fed to corresponding PE during each cycle period.

According to the rearranged data pipelining scheme, we then define the operand-vector $B_i^{'}$ and $V_i^{'}$ (for k number of PEs), where $0 \leq i \leq n-1$, to represent each operand cluster, whose initial value is

$$\begin{aligned} B_i^{'} &= [\ B_{(n-1-i)}, B_{(n-i)}, \cdots, B_{(n+k-2-i)}\ ] \\ V_i^{'} &= [\ V_{(n-1-i)},\ V_{(n-i)}, \cdots, V_{(n+k-2-i)}\ ] \end{aligned} \tag{3.8}$$

which can also be represented as

$$\begin{aligned} B_i^{'} &= [\ B_i^{(0)}, B_i^{(1)}, \cdots, B_i^{(k-2)}, B_i^{(k-1)}\ ] \\ V_i^{'} &= [\ V_i^{(0)}, V_i^{(1)}, \cdots, V_i^{(k-2)}, V_i^{(k-1)}\ ] \end{aligned} \tag{3.9}$$

then, the product C can be written as

$$C = \sum_{i=0}^{n-1} B_i^{'} V_i^{'}. \tag{3.10}$$

Then, we have the modified low register-complexity DL systolic multiplication algorithm as proposed in Algorithm 3.3.

Algorithm 3.3 Low Register-Complexity DL Systolic Multiplication.

Inputs: $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ are two field elements over $GF(2^m)$, where m is an odd number.

Output: $C = (c_0, c_1, \cdots, c_{m-1}) = A \cdot B$.

1. $C = 0$
2. $V = \overline{A} \gg 1$, where $\overline{A} = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0)$.
3. for $i = 0$ to $n - 1$ do.
4. $C_i = 0$.
5. $V_i' = [\ V_i^{(0)}, V_i^{(1)}, \cdots, V_i^{(k-2)}, V_i^{(k-1)}\ ] \gg id$.
6. $B_i' = [\ B_i^{(0)}, B_i^{(1)}, \cdots, B_i^{(k-2)}, B_i^{(k-1)}\ ] \gg id$.
7. for $j = 1$ to k do.
8. $C_i = (C_i \gg d) + L(V_i^{(j-1)}, B_i^{(j-1)})$.
9. end for.
10. $C = (C \gg kd) + C_i$.
11. end for.

---

where Steps 5 and 6 denote the initialization of the value of the operand-vector and their cyclic shifting. Through this arrangement, the register count used for pipelining shifted-operands can be removed.

3.2.3 Proposed DL Systolic GNB Multiplication Algorithm

Based on the discussion in Subsections A and B, we then combine the two modified algorithms together to propose our novel multiplication algorithm. The proposed multiplication algorithm for DL systolic GNB multiplier over $GF(2^m)$, which can achieve both low critical-path delay and low register-complexity, is proposed in Algorithm 3.4.

Algorithm 3.4 Proposed DL Systolic Multiplication.

Inputs: $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ are two field elements over $GF(2^m)$, where m is an odd number.

Output: $C = (c_0, c_1, \cdots, c_{m-1}) = A \cdot B$.

1. $C = 0$
2. $V = \overline{A} \gg 1$, where $\overline{A} = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0)$.
3. for $i = 0$ to $n - 1$ do.

4. $C_i = 0$.
5. $V_i' = [\ V_i^{(0)}, V_i^{(1)}, \cdots, V_i^{(k-2)}, V_i^{(k-1)}\ ] \gg id$.
6. $B_i' = [\ B_i^{(0)}, B_i^{(1)}, \cdots, B_i^{(k-2)}, B_i^{(k-1)}\ ] \gg id$.
7. $B_i^{(0)^*} = RAO(B_i^{(0)})$.
8. for $j = 1$ to k do.
9. $C_i = (C_i \gg d) + MA(V_i^{(j-1)}, B_i^{(j-1)^*})$.
10. $B_i^{(j)^*} = RAO(B_i^{(j)})$.
11. end for.
12. $C = (C \gg kd) + C_i$.
13. end for.

As one can see in Algorithm 3.4, Steps 5 and 6 perform the operations of initialization of the value of the operand cluster as well as the cyclic shifting. RAO is performed by Steps 7 and 10, and AC is computed by Step 12. By computing the first RAO in advance and rearranging the data broadcasting, we have successfully shorten the critical-path delay and reduced the register-complexity.

3.3 Proposed Low Critical-path Delay Low Register-Complexity DL Systolic GNB Multiplier

Based on the proposed Algorithm 3.4, we present here the proposed DL systolic GNB multiplier over $GF(2^m)$, which can achieve both low critical-path delay and low register-complexity.

3.3.1 Proposed Structure

The proposed DL systolic structure of GNB multiplier over $GF(2^m)$ based on the proposed Algorithm 3.4 is depicted in Fig. 3.8. As shown in Fig. 3.8(a), it consists of one AC, $(k+1)$ number of PEs, and two shift-registers for operands B and V, respectively. The detailed internal structures of AC and PEs are presented in Fig. 3.8(b). The shift-registers rearrange all the bits of operand B and V, so that there will be only one operand

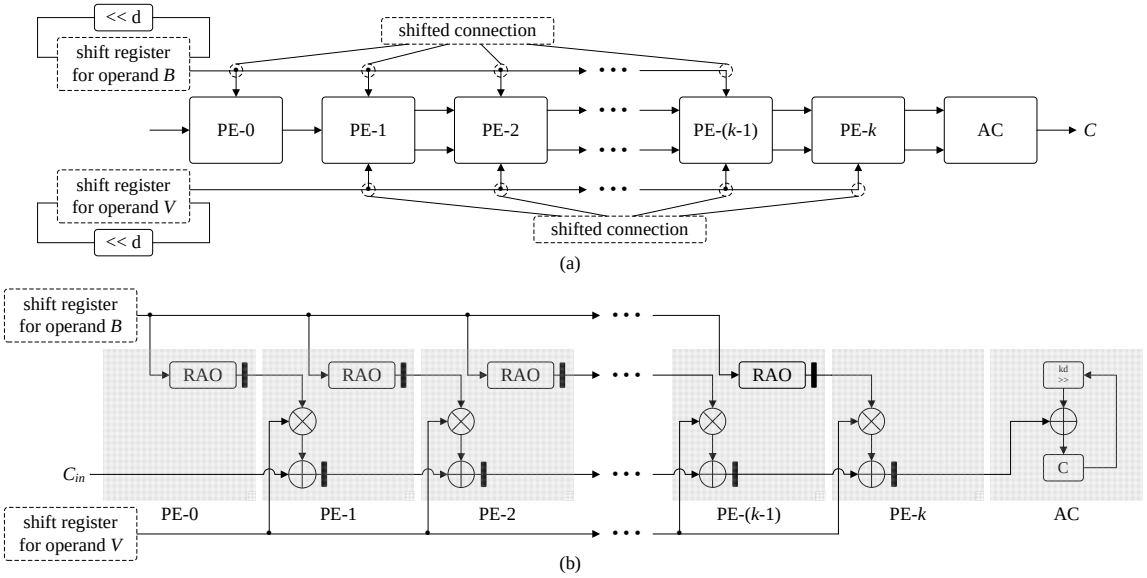

Figure 3.8: Proposed low critical-path delay low register-complexity DL systolic GNB multiplier over $GF(2^m)$. (a) Proposed structure of DL systolic GNB multiplier. (b) Detailed internal structures of PEs, where black boxes denote registers.

cluster to be fed to k number of PEs in one clock cycle period. For PE-0, there is only one component: RAO, which yields an output to PE-1 to perform multiplication with operand V and then the addition. The internal structures of PEs from PE-1 to PE-$(k-1)$ are the same, where each PE contains one RAO, one multiplication operation and one addition operation. The RAO performs reconstructed addition operation, whose result is yielded to the multiplication operation in the next PE on its right. The multiplication is then performed between operand V and the result of RAO from the previous PE, and the result from multiplication is then added with the output of addition operation from the previous PE. PE-k performs only multiplication and addition operations. The result of the last PE (PE-(k)) will fed to AC every cycle after the AC receives its first input from left, and the final result C can be obtained after $(n + k + 1)$ clock cycles.

3.3.2 An Example

Let us take type 4 GNB over $GF(2^7)$ as an example. Assume $d = 1$ and $k = 7$, then, we have $q = 7$ and $n = 1$. The multiplication matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{3.11}$$

Thus, we can obtain

$$\begin{aligned} c_0 = \; & a_0 b_1 + a_1(b_0 + b_2 + b_5 + b_6) \\ & + a_2(b_1 + b_3 + b_4 + b_5) + a_3(b_2 + b_5) \\ & + a_4(b_2 + b_6) + a_5(b_1 + b_2 + b_3 + b_6) \\ & + a_6(b_1 + b_4 + b_5 + b_6). \end{aligned} \tag{3.12}$$

According to Algorithm 3.4, (3.12) can also be represented as

$$c_0 = \; V_0^{(0)} \cdot B_0^{(0)^*}, \tag{3.13}$$

where

$$V_0^{(0)} = [\; a_0, a_1, a_2, a_3, a_4, a_5, a_6 \;], \tag{3.14}$$

and

$$\begin{aligned} B_0^{(0)^*} = [\; & b_1, (b_0 + b_2 + b_5 + b_6), \\ & (b_1 + b_3 + b_4 + b_5), (b_2 + b_5), \\ & (b_2 + b_6), (b_1 + b_2 + b_3 + b_6), \\ & (b_1 + b_4 + b_5 + b_6) \;]. \end{aligned} \tag{3.15}$$

Table 3.1: COMPARISON OF THE AREA AND TIME COMPLEXITIES FOR VARIOUS DL MULTIPLIERS OVER $GF(2^m)$

Design	#XOR	#AND	#Register	Latency	CPD
GNB [44]	$< \frac{d(m-1)}{2}(T-1)+dm$	dm	$3m$	$\lceil \frac{m}{d} \rceil$	T_{W1}
GNB [46], [47]	$\leq \frac{d(m-1)}{2}(T-1)+dm$	dm	$3m$	$\lceil \frac{m}{d} \rceil$	T_{W1}
PB [28]	$dm+2d$	dm	$4m+3d+1$	$2\lceil \frac{m}{d} \rceil$	T_{W2}
GNB [43]	$d^2+d+mT+1$	d^2	$3.5d^2+8mT$	$d+\frac{mT}{d}(\frac{mT}{d}+1)$	T_A+T_X
GNB [48]	$\leq \frac{\lceil \sqrt{\frac{m}{d}} \rceil d(m-1)}{2}(T-1)+(1+\lceil \sqrt{\frac{m}{d}} \rceil d)m$	$\lceil \sqrt{\frac{m}{d}} \rceil dm$	$(1+3\lceil \sqrt{\frac{m}{d}} \rceil)m$	$\leq 2\lceil \sqrt{\frac{m}{d}} \rceil$	T_{W1}
Proposed	$\leq \frac{\lceil \sqrt{\frac{m}{d}} \rceil d(m-1)}{2}(T-1)+(1+\lceil \sqrt{\frac{m}{d}} \rceil d)m$	$\lceil \sqrt{\frac{m}{d}} \rceil dm$	$< (3+2\lceil \sqrt{\frac{m}{d}} \rceil)m$	$\leq 2\lceil \sqrt{\frac{m}{d}} \rceil + 1$	T_{W3}

CPD: Critical-path delay.
$T_{W1} = T_A + (\lceil \log_2 T \rceil + \lceil \log_2 (d+1) \rceil) T_X$,
$T_{W2} = T_A + (\lceil \log_2 T \rceil + \lceil \log_2 d \rceil) T_X$,
$T_{W3} = T_A + (\lceil \log_2 (d+1) \rceil) T_X$,
T_A and T_X are the critical-path delay of AND gate and XOR gate,respectively.

We can also obtain the value for $V_0^{(1)} \cdots V_0^{(6)}$ and $B_0^{(1)^*} \cdots B_0^{(6)^*}$. Then, following the steps in Algorithm 3.4, we can finally get the partial product C_i, which is also the final product C in this case.

3.4 Area-Time Complexities

3.4.1 Theoretical Comparison

The area-time complexities of the proposed and the existing ones of [26, 43-44, 46-48] in terms of logic-gate-count, register-count, critical-path delay, and latency are shown in Table 3.1. According to the table, the latency of the proposed DL systolic multipliers is $(2\lceil \sqrt{m/d} \rceil + 1)$, while [26] requires $2\lceil m/d \rceil$ and a number of existing GNB multipliers require the latency of $\lceil m/d \rceil$. In addition, multiplier in [43] has latency of $(d + mT/d(mT/d + 1))$ clock cycles, which is more than the proposed one. We have also chosen the field of $GF(2^{409})$ to have a detailed comparison of the latencies of various DL GNB multipliers. As shown in Tables 3.2, the latency of our proposed structure is only 1 clock cycle more than the one in [48]. However, the latency of proposed one is much lower than the other existing multipliers. As the digit-size increases from 2 to 14, our proposed multiplier has nearly 2.3-13.2 times less latency compared with those in [26] and [44].

In terms of register-complexity and critical-path delay, one can see from Table 3.1 that

Table 3.2: COMPARISON OF LATENCY OVER $GF(2^{409})$

Digit-size (d)	2	4	6	8	10	12	14
[26]	410	206	138	104	82	70	60
[44]	205	103	69	52	41	35	30
[48]	30	22	18	16	14	12	12
Proposed	31	23	19	17	15	13	13

Table 3.3: COMPARISON OF CRITICAL-PATH DELAY WITH DIFFERENT DIGIT-SIZE FOR VARIOUS DL MULTIPLIERS OVER $GF(2^{409})$

Design	Digit-size	CPD
Proposed	7	$T_A + 3T_X$
	13	$T_A + 4T_X$
	19	$T_A + 5T_X$
	33	$T_A + 6T_X$
[48]	7	$T_A + 5T_X$
	13	$T_A + 6T_X$
	19	$T_A + 7T_X$
	33	$T_A + 8T_X$
[26]	7	$T_A + 5T_X$
	13	$T_A + 6T_X$
	19	$T_A + 7T_X$
	33	$T_A + 8T_X$

the DL-PIPO multiplier in [48] requires $(1 + 3\lceil\sqrt{m/d}\rceil)m$ registers and its critical-path delay is $T_{W1} = T_A + (\lceil\log_2 T\rceil + \lceil\log_2(d+1)\rceil)T_X$. While the proposed architecture requires $< (3+2\lceil\sqrt{m/d}\rceil)m$ registers, and the critical-path delay is $T_{W3} = T_A + (\lceil\log_2(d+1)\rceil)T_X$, which is significantly less than the existing one of [48]. We have also chosen field size of $GF(2^{409})$ to have a detailed comparison of ours and the existing ones of [26] and [48] as shown in Table 3.3. It is seen that the critical-path delay of our proposed structure is significantly less than the existing ones.

3.4.2 ASIC Implementation

We have also synthesized our proposed and the existing designs to obtain the area-time complexity. We have used Synopsys Design Compiler based on Taiwan Semiconductor Manufacturing Company (TSMC) 65-nm standard-cell library. The results in terms of area, critical-path delay, and latency-cycles (including latency time) of our proposed systolic structure are shown in Table 3.4 with different field sizes (m = 163, 283, and

Table 3.4: ASIC SYNTHESIS RESULTS OF THE PROPOSED SYSTOLIC MULTIPLIER

m, T	Area $[\mu m^2]$	CPD $[ns]$	d	Latency	Time $[ns]$
$163, 4$	$14,254$	0.87	9	11	9.57
	$16,146$	0.94	11	9	8.46
	$38,894$	1.96	28	7	13.72
$283, 6$	$109,512$	1.13	16	11	12.43
	$170,811$	1.31	21	9	11.79
	$310,254$	2.36	36	7	16.52
$409, 4$	$202,972$	1.04	13	13	13.52
	$260,882$	1.27	19	11	13.97
	$618,238$	2.67	41	9	24.03

Table 3.5: ASIC SYNTHESIS RESULTS FOR THE EXISTING AND THE PROPOSED DL MULTIPLIERS OVER $GF(2^{409})$

Design	Digit-size	Latency	Area $[\mu m^2]$	CPD $[ns]$	Latency time $[ns]$	Area-delay product (ADP) $[pm^2 s]$
Proposed	7	17	$123,742$	0.91	15.47	$1,914$
	13	13	$202,972$	1.04	13.52	$2,744$
	19	11	$260,882$	1.27	13.97	$3,644$
[48]	7	16	$128,201$	1.28	20.4	$2,615$
	13	12	$207,222$	1.38	16.6	$3,439$
	19	10	$264,327$	1.63	16.3	$3,782$
[46], [47]	13	32	$71,760$	1.44	46.1	$3,308$
	18	23	$115,806$	1.55	35.6	$4,122$
	23	18	$147,475$	1.68	29.7	$4,380$
[26]	13	64	$58,917$	1.23	78.7	$4,637$

409) and digit sizes. As shown in Table 3.4, our proposed design performs better when the digit size becomes smaller.

We have also compared our systolic multiplier with the existing DL multipliers in terms of different digit sizes with the same field size $m = 409$, as shown in Table 3.5. One can see that for the same digit size $d = 13$, our proposed multiplier has shorter critical-path delay and smaller latency time compared with the other multipliers. Moreover, the area-delay product (ADP) of the proposed one is the smallest among all the multipliers in Table 3.5. For the field size of $m = 409$ (digit size of $d = 13$), our proposed systolic multiplier has a latency time of $13.52ns$ and area of $202,972\mu m^2$, while the multiplier in [48] (with the same digit size) needs $16.6ns$ and has $207,222\mu m^2$ ([46] and [47] have even larger results). For digit size of $d = 7$, the ADP of the proposed structure is 26.7% more efficient than [48]. For $d = 13$, our design is 18.6%, 70.7%, and 82.8% faster than that (total time) in [48], [46] ([47]), and [26], respectively. The ADP of the proposed design with $d = 13$ is 20.2%, 17.0%, and 40.8% less the existing of [48], [46] and [47], and [26], respectively.

Chapter 4

Conclusion

4.1 Low complexity hybrid-size systolic polynomial multipliers

An efficient, new hybrid structure of pentanomials and trinomials for low-complexity implementation of finite field multipliers over $GF(2^m)$ has been proposed in Chapter 2. Based on the similarities of the computation matrices of operand A of the pentanomial-based and trinomial-based multipliers, we sperate the common bits of operand A of both multipliers to form a new matrix named AOP-liked matrix, which reduces the register-complexity. Moreover, a novel low register-complexity algorithm for hybrid systolic finite field polynomial multipliers has been presented. We have also introduced structures for low-latency and digit-parallel implementation. Both the theoretical analysis and the FPGA implementation results have confirmed the higher efficiency of the proposed architectures compared with the competing ones.

4.2 Low complexity digit-level systolic Gaussian normal basis multipliers

A low critical-path delay, low register-complexity DL systolic GNB multiplier over $GF(2^m)$ has been proposed in the third chapter of this thesis. We have proposed a novel multiplication algorithm to reduce the critical-path delay and the register-complexity. Moreover, both theoretical and ASIC implementation results are presented for comparison. Based

on our presented results, our proposed design has smaller critical-path delay and fewer register-complexity when compared with the existing DL systolic multipliers. The proposed DL multiplier, thus, can be extended and employed in sensitive usage models including high-performance cryptographic applications.

Chapter 5

Publication

Q. Shao, Z. Hu, S. Chen, P. Chen, R. Azarderakhsh, M. Mozaffari-Kermani, and J. Xie, "Low Critical-Path Low-Complexity Digit-Level Systolic Gaussian Normal Basis Multiplier,"IEEE Trans. VLSI Systems, submitted for review.

Chapter 6

Reference

[1] 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.

[2] N. R. Murthy and M. N. S. Swamy, "Cryptographic applications of Brahmaqupta-Bha skara equation," *IEEE Trans. Circuits and Systems-I*, vol. 53, no. 7, pp. 1565-1571, 2006.

[3] M. Sun, L. E. Burke, Z.-H. Mao, Y. Chen, H.-C. Chen, Y. Bai, Y. Li, C. Li, and W. Jia. "eButton: A wearable computer for health monitoring and personal assistance," in *Proc. Design Automation Conference*, pp. 1-6, 2014.

[4] D. Schinianakis and T. Stouraitis, "Multifunction residue architectures for cryptography," *IEEE Trans. Circuits and Systems-I*, vol. 61, no. 4, pp. 1156-1169, 2014.

[5] National Institute of Standards and Technology, "FIPS 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2," 2000.

[6] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* New York: Wiley, 1999.

[7] 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.*, vol. 54, no. 9, pp. 1061-1070, 2005.

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

[9] J. Xie, P. K. Meher, and J. He, "Low-latency area-delay-efficient systolic multiplier

over $GF(2^m)$ for a wider class of trinomials using parallel register sharing," in *Proc. IEEE Int. Sym. Circuits and Systems*, pp. 89-92, 2012.

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

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

[12] P. Montgomery, "Modular multiplication without trial division," *Math, Computation*, vol. 44, no. 170, pp. 519-521, 1985.

[13] R. Azarderakhsh, K. Jarvinen, and V. Dimitrov, "Fast inversion in $GF(2^m)$ with normal basis using hybrid-double multipliers," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 1041-1047, 2014.

[14] R. Azarderakhsh, D. Jao, and H. Lee, "Space complexity reduction algorithms for Gaussian normal basis multiplication," *IEEE Trans. Information Theory*, vol. 61, no. 5, pp. 2357-2369, 2015.

[15] R. Azarderakhsh and M. Mozaffari-Kermani, "High-performance two-dimensional finite field multiplication and exponentiation for cryptographic applications," *IEEE Trans. Computer Aided Design Integrated Circuits Systems*, vol. 34, no. 10, pp. 1-8, 2015.

[16] C-Y. Lee and P. K. Meher, "Area-efficient subquadratic space-complexity digit-serial multiplier for type-II optimal normal basis of $GF(2^m)$ using symmetric TMVP and block recombination techniques," *IEEE Trans. Circuits and Systems-I*, vol. 62, no. 12, pp. 2846-2855, 2015.

[17] S. Talapatra, H. Rahaman, and S. K. Saha, "Unified digit serial systolic Montgomery multiplication architecture for special classes of polynomials over $GF(2^m)$," in *Proc. Conf. Digital System Design: Architectures, Methods and Tools*, pp. 427-432, 2010.

[18] S. Fenn and M. Parker, "Bit-serial multiplication in $GF(2^m)$ using all-one polynomials," *IEE proc. Com. Digit. Tech.*, vol. 144, no. 6, pp. 391-393, 1997.

[19] 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. Comput.*, vol. 54, no. 12, pp. 1628-1629, 2005.

[20] H.-S. Kim and S.-W. Lee, "LFSR multipliers over $GF(2^m)$ defined by all-one poly-

nomial," *Integr., the VLSI J.*, vol. 40, no. 4, pp. 571-578, 2007.

[21] P. K. Meher and C.Y. Lee, "An optimized design of serial-parallel finite field multiplier for $GF(2^m)$ based on all-one polynomials," in *Proc. ASP-DAC*, pp. 210-215, 2009.

[22] M.-Sandoval, M. F.-Uribe, and C. Kitsos, "Bit-serial and digit-serial $GF(2^m)$ Montgomery multipliers using linear feedback shift registers," *IET Comput. & Digital Tech.*, vol. 5, no. 2, pp. 86-94, 2011.

[23] 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.*, vol. 50, no. 6, pp. 385-393, 2001.

[24] J. Xie, P. K. Meher, and J. He, "Low-complexity multiplier for $GF(2^m)$ based on all one polynomials," *IEEE Trans. VLSI Systems*, vol. 21, no. 1, pp. 168-172, 2013.

[25] Y.-R. Ting, E.-H. Lu, and Y.-C. Lu, "Ringed bit-parallel systolic multipliers over a class of fields $GF(2^m)$," *Integr., the VLSI J.*, vol. 38, no. 4, pp. 571-578, 2005.

[26] 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.*, vol. 18, no. 5, pp. 847-852, 2010.

[27] T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of fields $GF(2^m)$," *Information and Computation*, vol. 83, no. 1, pp. 21-40, 1989.

[28] J. L. Imana, R. Hermida, and F. Tirado, "Low complexity bit-parallel multipliers based on a class of irreducible pentanomials," *IEEE Trans. VLSI Sys.*, vol. 14, no. 12, pp. 1388-1393, 2006.

[29] J. Xie, P. Meher, and Z. Mao, "Low-latency high-throughput systolic multipliers over $GF(2^m)$ for NIST recommended pentanomials," *IEEE Trans. Circuits and Sys.-Regular Papers*, vol. 62, no. 3, pp. 881-890, 2015.

[30] P. Chen, S. Basha, M. Mozaffari-Kermani, R. Azarderakhsh and J. Xie, "FPGA realization of low register systolic all-one-polynomial multipliers over $GF(2^m)$ and their applications in trinomial multipliers," *IEEE Trans. VLSI Sys.*, vol. PP, no. 3, pp. 1-10, 2016.

[31] R. R. Farashahi and M. Joye, "Efficient arithmetic on Hessian curves," in *Proc. Int. Conf. Pract. Theory Public Key Cryptograph.*, pp. 243-260, 2010.

[32] Ç. K. Koç and B. Sunar, "An efficient optimal normal basis Type II multiplier over

$GF(2^m)$," *IEEE Trans. Comput.*, vol. 50, no. 1, pp. 83-87, 2001.

[33] J. Xie, P. K. Meher, and Z. Mao, "High-throughput digit-level systolic multiplier over $GF(2^m)$ based on irreducible trinomials," *IEEE Trans. Circ. and Syst.-II*, vol. 62, no. 5, pp. 481-485, 2015.

[34] J. Adikari, V. S. Dimitrov, and R. J. Cintra, "A new algorithm for double scalar multiplication over Koblitz curves," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 709-712, 2011.

[35] K. Järvinen and J. Skyttä, "On parallelization of high-speed processors for elliptic curve cryptograph," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 9, pp. 1162-1175, 2008.

[36] C. Lee, P. Meher, and J. Patra, "Concurrent error detection in bit-serial normal basis multiplication over $GF(2^m)$ using multiple parity prediction schemes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1234-1238, 2010.

[37] W. Geiselmann and D. Gollmann, "Symmetry and duality in normal basis multiplication," in *Proc. Sixth Symp. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC)*, pp. 230-238, 1989.

[38] R. Azarderakhsh, and A. Reyhani-Masoleh, "Efficient FPGA implementation of point multiplication on binary Edwards and generalized Hessian curves using Gaussian normal basis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1453-1466, 2012.

[39] 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.*, vol. 17, no. 6, pp. 747-757, 2009.

[40] S. Kwon, "A low complexity and a low latency bit parallel systolic multiplier over $GF(2^m)$ using an optimal normal basis of type II," in *Proc. 16th IEEE Symp. Comput. Arithmetic*, pp. 196-202, 2003.

[41] J. Fan, D. V. Batina, T. Guneysu, C. Paar, and I. Verbauwhede, "Breaking elliptic curves cryptosystems using reconfigurable hardware," in *Proc. Int. Conf. Field Program. Logic Appl.*, pp. 133-138, 2010.

[42] A. Wang and S. Fan, "Efficient montgomery-based semi-systolic multiplier for even-type GNB of $GF(2^m)$," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 415-419, 2012.

[43] C. Lee and C. W. Chiou, "Scalable Gaussian normal basis multipliers over $GF(2^m)$

using Hankel matrix-vector representation," *J. Signal Process. Syst.*, vol. 69, no. 2, pp. 197-211, 2012.

[44] R. Azarderakhsh and A. Reyhani-Masoleh, "A modified low complexity digit-level Gaussian normal baasis multiplier," in *Proc. 3rd Int. Workshop Arithmetic Finite Fields*, pp. 25-40, 2010.

[45] *Digital Signature Standard*, National Institute of Standards and Technology, Gaithersburg, MD, USA, Jan. 2000.

[46] A. Reyhani-Masoleh, "Efficient algorithms and architectures for field multiplication using Gaussian normal bases," *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 34-47, 2006.

[47] S. Kwon, K. Gaj, C. H. Kim, and C. P. Hong, "Efficient linear array for multiplication in $GF(2^m)$ using a normal basis for elliptic curve cryptography," in *Proc. 6th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, pp. 76-91, 2014.

[48] R. Azarderakhsh, M. Mozaffari Kermani, S. Bayat-Sarmadi, and C.-Y. Lee, "Systolic Gaussian normal basis multiplier architectures suitable for High-performance applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1969-1972, 2015.

[49] IEEE Std 1363-2000, "IEEE Standard Specification for Public-Key Cryptography," Jan. 2010.

[50] US Dept. of Commerce/NIST, "National Institute of Standards and Technology," *Digital Signature Standard, FIPS Publications 186-2*, Jan. 2010.

[51] J. Massey and J. Omura, *Computational Method and Apparatus for Finite Arithmetic*, US Patent 4587627, Washington, D.C., 1986.

[52] T. Beth and D. Gollmann, "Algorithm engineering for public key algorithms," *IEEE J. Selected Areas in Communications*, vol. 7, no. 4, pp. 458-466, 1989.

[53] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone, "An implementation for a fast public-key cryptosystem," *J. Cryptology*, vol. 3, no. 2, pp. 63-79, 1991.

[54] A. Reyhani-Masoleh and M. A. Hasan, "Efficient digit-serial normal basis multipliers over binary extension fields," *ACM Trans. Embedded Computing Systoms*, vol. 3, no. 3, pp. 575-592, 2004.

[55] A. H. Namin, H. Wu, and M. Ahmadi, "A word-level finite field multiplier using normal basis," *IEEE Trans. Comput.*, vol. 60, no. 6, pp. 890-895, 2006.

[56] C. Lee and P. Chang, "Digit-serial Gaussian normal basis multiplier over $GF(2^m)$ using Toeplitz Matrix-approach," in *Proc. Int. Conf. Computational Intelligence and Software Eng. (CiSE)*, pp. 1-4, 2009.