Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases

Reza Azarderakhsh, *Student Member*, *IEEE*, and Arash Reyhani-Masoleh, *Member*, *IEEE*

Abstract—The extensive rise in the number of resource constrained wireless devices and the needs for secure communications with the servers imply fast and efficient cryptographic computations for both parties. Efficient hardware implementation of arithmetic operations over finite field using Gaussian normal basis is attractive for public key cryptography as it provides free squarings. In this paper, we first present two low-complexity digit-level multiplier architectures. It is shown that the proposed multipliers outperform the existing Gaussian normal basis (GNB) multiplier structures available in the literature. Then, for the first time, using these two architectures, we propose a new digit-level hybrid multiplier which performs two successive multiplications with the same latency as the one for one multiplication. We have studied the efficiency of the proposed hybrid architecture in terms of area and time delay for different digit sizes. The main advantage of this new hybrid architecture is to speed up exponentiation and point multiplication whenever double-multiplication is required and the traditional schemes fail due to the data dependencies. We have investigated the applicability of the proposed hybrid structure to reduce the latency of exponentiation-based cryptosystems. Our analysis and timing results show that the expected acceleration in double-exponentiation is considerable. Prototypes of the presented low-complexity multiplier architectures and the proposed hybrid architecture are implemented and experimental results are presented.

Index Terms—Cryptosystems, Gaussian normal basis, double-multiplication, digit-level multiplier, double-exponentiation

1 Introduction

Information security in resource-constrained environments (such as smart cards and RFID tags) and high-performance web server applications (such as secure e-;commerce transactions and online banking) highly requires efficient cryptographic computations. The former applications are suffering from availability of silicon area, while the latter ones are suffering from low speed of the current security protocols. The arithmetic operations in the finite fields over characteristic two $GF(2^m)$ are largely utilized for cryptographic algorithms such as point multiplication in elliptic curve cryptography (ECC) [1], [2] and exponentiation-based cryptosystems [3], [4]. The exponentiation is an important arithmetic operation for public key cryptosystems such as ElGamal encryption scheme [3] and Diffie-Hellman key agreement [4]. It mainly requires successive field multiplications and squarings and its efficiency relies on the computation of these operations and the representation of field elements.

A finite field can be represented using different bases such as polynomial (or standard) basis, normal basis, and dual basis. Among them, normal basis is more efficient in hardware implementations since squaring of a field element over $GF(2^m)$ can be performed by a simple cyclic shift. This makes normal basis more attractive for the cryptosystems that utilize frequent squarings (e.g., point multiplication on Koblitz curves and exponentiation-based cryptosystems). Gaussian normal basis (GNB) [5], is a special class of normal basis and has received considerable attention in the literature for its low complexity. GNB is included in various standards such as IEEE [6] and NIST [7] for elliptic curve digital signature algorithm (ECDSA). The implementation of finite field multipliers using normal basis and more specifically GNB can be categorized, in terms of their structures, into three groups: 1) bit-level which includes: parallel-in serial-out (PISO) [8], serial-in parallel-out (SIPO) [9], [10], [11], and parallel-in parallel-out (PIPO) [12], [13], 2) digit-level including the structures of: parallel-in serial-out [14], parallel-in parallel-out [15], [16], [17], and serial-in parallel-out [18], and 3) bit-parallel which includes: [19], [20], and [21] multipliers.

Bit-level multipliers provide the lowest possible area complexity. The first bit-level normal basis multiplier has been invented by Massey and Omura [8] in which all coordinates of both input operands should be presented during multiplication operation. Bit-level SIPO multipliers have been studied for normal basis and two different structures, namely Least Significant Bit (LSB) first and Most Significant Bit (MSB) first structures, have been proposed by Beth and Gollmann in [10].

Digit-level multipliers are alternatives for bit-level and bit-parallel multipliers in which the digit size can be chosen depending on the amount of the resources available. A digit-level PIPO version of Massey-Omura multiplier [22] and its improved version [15] are used in ECC-based crypto-processors in [23] and [24]. It has been mentioned that in order to satisfy high-speed and low-complexity

• *The authors are with the Department of Electrical and Computer Engineering, The University of Western Ontario, 1151 Richmond Street North, London, ON N6A 5B8, Canada.*
E-mail: {razarder, areyhani}@uwo.ca.

Manuscript received 18 July 2011; revised 29 Nov. 2011; accepted 30 Dec. 2011; published online 16 Jan. 2012.
Recommended for acceptance by A. Nannarelli.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-07-0488.
Digital Object Identifier no. 10.1109/TC.2012.22.

requirements of cryptographic applications, there is a need to design efficient architectures for finite field multiplication using normal basis. In [16], two efficient digit-level PISO and PIPO GNB multipliers are presented. In [25], a subexpression sharing algorithm is introduced to obtain the least number of gates for the digit-level PIPO multiplier. In the following, we summarize the contributions of this work.

1.1 Our Contributions

In this paper, we present a new digit-level hybrid multiplier which performs two multiplications together (double-multiplication) with the same number of clock cycles required as the one for one multiplication. It has advantages for high-speed finite field arithmetic operations such as exponentiation and elliptic curves point multiplication. The hybrid structure is developed by connecting the output of the proposed digit-level PISO GNB multiplier into the input of a new digit-level SIPO multiplier. For the digit-level PISO GNB multiplier, we first use a subexpression elimination algorithm [25], to reduce its area complexity. Then, we propose a low-complexity and fast least significant digit (LSD)-first digit-level SIPO GNB multiplier based on the bit-level SIPO multiplier presented in [10]. The complexities of these multipliers are derived and compared with the counterparts. It is shown that the presented improved LSD-first digit-level SIPO multiplier outperforms its counterparts in terms of both time and area complexities.

To the best of the authors' knowledge, this is the first digit-level hybrid GNB multiplier which performs two multiplications with the same latency as the one required for one multiplier proposed in the literature. In order to investigate the applicability of the proposed hybrid multiplier architecture, we employ it for double-exponentiation which is the key operation for Schnorr [26] and ElGamal-type signature verification algorithms [3]. We further note that this scheme can be incorporated to reduce the latency of point multiplication for ECC-based cryptosystems when other schemes (such as parallelization and interleaving) fail due to data dependencies. To obtain the actual implementation results, the proposed multiplier architectures are coded using VHDL and then implemented on both Xilinx Virtex-4 field-programmable gate array (FPGA) and 65-nm CMOS application-specific integrated circuit (ASIC) technology (synthesized) for different digit sizes.

The organization of this paper is as follows: in Section 2, we review preliminaries of multiplication in Gaussian normal basis over $GF(2^m)$. In Section 3, an improved digit-level SIPO architecture for GNB multiplication is presented and its area complexity reduced. Also, we present a low-complexity digit-level PISO multiplier architecture in this section. In Section 4, a new hybrid structure, which is composed of a digit-level PISO multiplier and a digit-level SIPO multiplier, is presented. We also present the application of such hybrid structure in this section. In Section 5, the performance of the proposed structures are investigated by implementing each multiplier as well as the hybrid structure on FPGA and ASIC. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we present the definition of Gaussian normal basis and briefly explain bit-level and digit-level multiplier architectures.

2.1 Multiplication Using Gaussian Normal Basis

GNB has been constructed by Ash et al. [5] and is a special class of normal basis which is included in the IEEE 1363 [6] and NIST [7] standards for ECDSA and exists for every $m > 1$ that is not divisible by eight [27].

Definition 1 ([27]). *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 $N = \{\beta, \beta^2, \ldots, \beta^{2^{m-1}}\}$ over $GF(2^m)$ is called the Gaussian normal basis (GNB) of type T, $T > 1$.*

The complexities of type T GNB multiplier in terms of time and area depend on $T > 1$. In this paper, we only consider the GNBs with odd values of m which implies that T is an even number.

Let $A = (a_0, a_1, \ldots, a_{m-1}) = \sum_{i=0}^{m-1} a_i \beta^{2^i}$ and $B = (b_0, b_1, \ldots, b_{m-1}) = \sum_{j=0}^{m-1} b_j \beta^{2^j}$ be two field elements over $GF(2^m)$ and assume $C \in GF(2^m)$ be their product, i.e., $C = (c_0, c_1, \ldots, c_{m-1}) = AB$. Then, the first coordinate of C, i.e., c_0 can be obtained from an explicit formula given in [6] as follows:

$$\begin{aligned} c_0 &= a_0 b_1 + \sum_{k=2}^{p-2} a_{F(k)} b_{F(k+1)} \\ &= a_0 b_1 + \sum_{i=1}^{m-1} a_i \left(\sum_{F(k)=i} b_{F(k+1)} \right), \quad 2 \le k \le p-2, \end{aligned} \tag{1}$$

where in (1), the sequence $F(1), F(2), \ldots, F(p-1)$ can be obtained by precomputation using

$$F(k) = F(2^i u^j \bmod p) = i,\ 1 \le i \le m-1,\ 0 \le j < T, \tag{2}$$

where u is an integer of order T mod p and $p = Tm + 1$ [6]. It is noted that for each i, $1 \le i \le m-1$, $F(k+1)$, $2 \le k \le p-2$ in (1), can be used as entries of a $(m-1) \times T$ matrix $\mathbf{R}$. Let us denote the (i,j)th element of this matrix as $R(i,j)$, $0 \le R(i,j) \le m-1$, $1 \le i \le m-1$, $1 \le j \le T$. Each row of the matrix $\mathbf{R}$, contains T entries of integer in $[0, m-1]$. Then, one can write c_0 as [16]

$$c_0 = a_0 b_1 + \sum_{i=1}^{m-1} a_i \left(\sum_{j=1}^{T} b_{R(i,j)} \right). \tag{3}$$

Note that, to obtain the lth coordinates of C, i.e., c_l one needs to add "l mod m" to all indices in (3).

Remark 1. From (2) one can realize that for $T > 2$ there are situations (for example, $F(k) = \frac{m-1}{2}$ and $F(k) = \frac{m+1}{2}$ for $T = 4$) where matrix $\mathbf{R}$ contains (two) equal entries.

2.2 Bit-Level GNB Multiplication

Massey and Omura (MO) [8] and Beth and Gollmann in [10], respectively, proposed bit-level PISO and bit-level SIPO multiplier architectures for normal basis multiplication. The former generates every bit of the multiplication in

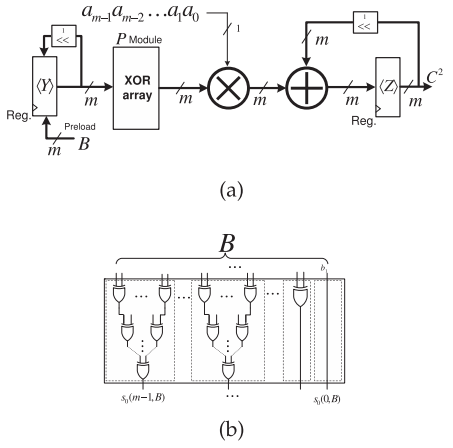


Fig. 1. (a) The architecture of LSB-first bit-level normal basis multiplier with parallel output [10]. (b) The architecture of P module for type T GNB.

each clock cycle, whereas the latter provides all output coordinates in parallel after m clock cycles. There are two type of bit-level SIPO multipliers named least significant bit (LSB)-first and most significant bit (MSB)-first. In the following, we review the LSB-first SIPO multiplier architecture using GNB. In this paper, this architecture is extended to digit level and then the area complexity of the digit-level multiplier is reduced. In an LSB-first bit-level multiplication, having all elements of one operand, say B, to be present, the other operand, i.e., A, is processed from its LSB, i.e., a_0, and in each clock cycle one bit is processed.

Lemma 1 ([10]). *Let A and B be two elements of $GF(2^m)$ and C be their multiplication, i.e., $C = AB$, then similar to Horner's rule one can obtain*

$$C = \Big(\big(\cdots\big(\big(a_0\beta B\big)^{2^{-1}} + a_1\beta B^{2^{-1}}\big)^{2^{-1}} + \cdots\big)^{2^{-1}} + a_{m-1}\beta B^{2^{-(m-1)}}\Big)^{2^{-1}}. \tag{4}$$

Let us denote $P(B) = \beta B \in GF(2^m)$ as a field element in GNB. In [16], $P(B)$ is obtained for a PIPO GNB multiplier and similarly one can obtain it for SIPO GNB multiplier-based on the $\mathbf{R}$ matrix as

$$P(B) = (b_1, s_0(1,B), s_0(2,B), \ldots, s_0(m-1,B)), \tag{5}$$

where $s_0(i,B) = \sum_{j=1}^{T} b_{R(i,j)} \in \{0,1\}$, $1 \le i \le m-1$. Then using (5) and Lemma 1, we can state the following.

Corollary 1. *For GNB, the product of $A = (a_0, a_1, \ldots, a_{m-1}) \in GF(2^m)$, given in bit-serial fashion, and $B \in GF(2^m)$ can be written as*

$$C = ((\cdots((a_0P(B)) \ll 1 + a_1P(B \ll 1)) \ll 1 + \cdots) \ll 1 + a_{m-1}P(B \ll m-1)) \ll 1, \tag{6}$$

where $B^{2^{-i}}$ is realized by $B \ll i$ which denotes i-fold left cyclic shift of the coordinates of B.

Equation (6) can be realized by an architecture depicted in Fig. 1a. The implementation of $P(B) \in GF(2^m)$ given in (5) is performed by a P module shown in Fig. 1b for type T GNB. The product of $a_iP(B)$ in Fig. 1a denotes bitwise AND operation between a_i and elements of $P(B)$ and is performed using m 2-input AND gates. Also the sum

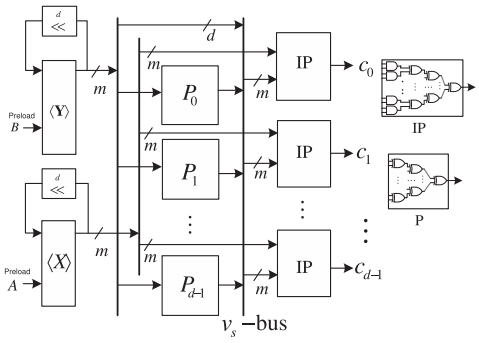


Fig. 2. The architecture of the digit-level PISO GNB multiplier [16].

(adder block in Fig. 1a) is implemented using m 2-input XOR gates. As one can see from Fig. 1a all bits of the operand B are available, while the coordinates of the operand A should be available in serial fashion with the LSB first, i.e, a_0. In this architecture, both m-bit registers $\langle Y\rangle = \langle y_0, y_1, \ldots, y_{m-1}\rangle$ and $\langle Z\rangle = \langle z_0, z_1, \ldots, z_{m-1}\rangle$ should be initialized with operand $B = (b_0, b_1, \ldots, b_{m-1})$ and $0 = (0, 0, \ldots, 0)$ (i.e., $Y(0) = B$ and $Z(0) = 0$), respectively. Let $Z(0)$ denote the initial value of the register $\langle Z\rangle$ and $Z(i)$, $1 \le i \le m$, be the content of the register $\langle Z\rangle$ in the clock cycle i. After one clock cycle the content of $\langle Z\rangle$ is $Z(1) = a_0P(B) \in GF(2^m)$. Then, the registers $\langle Y\rangle$ and $\langle Z\rangle$ are cyclically shifted to the left according to (6). As one can verify, after mth clock cycle the register $\langle Z\rangle$ contains the coordinates of $Z(m) = C^2 = (c_{m-1}, c_0, c_1, \ldots, c_{m-2})$ (see (6)). Thus, C can be obtained by a left cyclic shift of register $\langle Z\rangle$, i.e., $C = (Z(m) \ll 1)$. The presented architecture requires at most $(T-1)(m-1)$ XOR gates in the P module, m XOR gates for the adder, m AND gates, and two m-bit registers. Also, its critical-path delay due to delays through the P module ($\lceil \log_2 T\rceil T_X$), AND gates ($T_A$), and XOR gates ($T_X$) is $T_A + (1 + \lceil \log_2 T\rceil)T_X$.

2.3 Digit-Level PISO GNB Multiplier

In [16], a digit-level PISO GNB multiplier architecture is proposed. This architecture is depicted in Fig. 2 which requires both input operands A and B be available through multiplication. Assume the registers $\langle X\rangle$ and $\langle Y\rangle$ are preloaded with the operands A and B, respectively. Then, the formulations to perform multiplication are stated as follows based on (5).

Lemma 2. *For a digit-level multiplier architecture one needs to implement all d entries of $P(B)$ as*

$$P_l(B) = \big(b_{l+1 \bmod m}, S_0^l(1,B), S_0^l(2,B), \ldots, S_0^l(m-1,B)\big), \tag{7}$$

where $0 \le l \le d-1$ and

$$S_0^l(i,B) = \sum_{j=1}^{T} b_{l+R(i,j) \bmod m}, 1 \le i \le m-1, \tag{8}$$

and as both operand A and B are fully available, one can obtain the multiplication output as

$$c_l = a_l b_{l+1 \bmod m} + \sum_{i=1}^{m-1} a_{l+i \bmod m}\left(\sum_{j=1}^{T} b_{l+R(i,j) \bmod m}\right), \tag{9}$$

and implement d copies of c_l in hardware to achieve a digit-level architecture for l, $0 \le l \le d-1$. Then the registers $\langle X \rangle$ and $\langle Y \rangle$ should be d-fold cyclically shifted to the left to obtain the consecutive d coordinates of $C = AB$. This multiplier requires $q = \lceil \frac{m}{d} \rceil$, $1 \le q \le m$, $1 \le d \le m$, clock cycles to generate all the m coordinates of $C = AB$.

In the architecture which realizes (7) and (9), a d-fold left cyclic shift is denoted by "$\overset{d}{\ll}$" in Fig. 2.

3 Low-Complexity Architectures for Digit-Level GNB Multipliers

In this section, we first extend the bit-level SIPO multiplier presented in Section 2.2 to propose a low-complexity LSD-first digit-level SIPO (DL-SIPO) GNB multiplier. Then, we propose an improved low-complexity architecture for the digit-level PISO (DL-PISO) GNB multiplier presented in Section 2.3.

3.1 An Improved Architecture for Digit-Level SIPO GNB Multiplier

In a digit-level SIPO multiplier, the bits of an operand are grouped into digits and in each clock cycle one digit is processed. We extend the architecture of the LSB-first bit-level GNB multiplier architecture presented in Section 2.2 and propose a low-complexity LSD-first digit-level SIPO multiplier architecture. In the following, we present formulation, architecture, and complexity of the proposed multiplier architecture.

3.1.1 Formulation

Let us assume $A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, a_1, \ldots, a_{m-1})$, then one can group the bits into $q = \lceil \frac{m}{d} \rceil$ digits denoted by A_i, $0 \le i \le q-1$ as $(a_0, \ldots, a_{d-1})$ for the first digit followed by $(a_d, \ldots, a_{2d-1})$ for the second digit and finally $(a_{d(q-1)}, \ldots, a_{m-1})$ for the qth digit where d, $2 \le d \le m-1$, is denoted as the number of bits in each digit. Note that if the last digit does not have d bits, it will be appended by zeros at its most significant bit ends. Then, each digit can be represented as

$$\begin{aligned} A_i &= (a_{id}, a_{id+1}, \ldots a_{id+d-2}, a_{id+d-1}) \\ &= \sum_{j=0}^{d-1} a_{j+id} \beta^{2^j}, A_i \in GF(2^m) \end{aligned}$$

with respect to the GNB and thus operand A can be written as

$$A = \sum_{i=0}^{q-1} A_i^{2^{id}} = (A_0, A_1, \ldots, A_{q-1}).$$

Therefore, one can write their product $AB = C \in GF(2^m)$ as [28]

$$\begin{aligned} C &= AB \\ &= \sum_{i=0}^{q-1} A_i^{2^{id}} \cdot B = \sum_{i=0}^{q-1} (A_i \cdot B^{2^{-id}})^{2^{id}} = \sum_{i=0}^{q-1} (C^{(i)})^{2^{id}}, \end{aligned} \tag{10}$$

where

$$C^{(i)} = A_i B^{2^{-id}}. \tag{11}$$

In order to derive a formulation for multiplication whose implementation is more hardware-oriented we state the following.

Corollary 2. *Given the ith digit of A, i.e., A_i with d bits and a field element $B^{2^{-id}} \in GF(2^m)$, their product $C^{(i)} \in GF(2^m)$ can be obtained as*

$$C^{(i)} = \sum_{j=0}^{d-1} J^{2^j}\left(a_{j+id}, B^{2^{-(id+j)}}\right),$$

where $J(x, Y) = x \cdot P(Y) \in GF(2^m)$.

Proof. Using (11), one has

$$C^{(i)} = \sum_{j=0}^{d-1} a_{j+id} \beta^{2^j} \cdot B^{2^{-id}} = \sum_{j=0}^{d-1} (a_{j+id} \cdot \beta B^{2^{-id-j}})^{2^j}. \tag{12}$$

Now we define $J(x, Y)$ as a function of the product of a bit $x \in GF(2)$ and a field element $P(Y) \in GF(2^m)$ as

$$J(x, Y) = x \cdot P(Y). \tag{13}$$

Then, using (5) and Corollary 1, one can write $\beta B = P(B)$ to simplify $C^{(i)}$ in (12) as follows:

$$\begin{aligned} C^{(i)} &= \sum_{j=0}^{d-1} (a_{j+id} \cdot P(B \ll (id+j)))^{2^j} \\ &= \sum_{j=0}^{d-1} J^{2^j}(a_{j+id}, B^{2^{-(id+j)}}). \end{aligned} \tag{14}$$

This completes the proof. □

Then, the multiplication of A and B can be obtained from

$$C = AB = \sum_{i=0}^{q-1} (C^{(i)} \gg id). \tag{15}$$

In the following, we present the architecture of the proposed DL-SIPO GNB multiplier.

3.1.2 New Architecture

In order to map the formulation obtained in the previous section to hardware, an architecture for the LSD-first DL-SIPO GNB multiplier is depicted in Fig. 3. Initially, the register $\langle Y \rangle$ is loaded by $B = (b_0, b_1, \ldots, b_{m-1})$ and the register $\langle Z \rangle$ is cleared to 0. The d-fold left cyclic shifts are realized by "$\ll^d$" as shown in Fig. 3. Also, as one can see in this figure, the last digit of operand A, i.e., A_{q-1}, is appended by $r = qd - m$, $0 \le r \le d-1$, zeros at its most significant bit ends.

The DL-SIPO GNB multiplier architecture, has several P blocks shown as p_0 to p_{d-1} in Fig. 3a as a P array inside the Q block. As shown in this figure, P blocks use the shifted combination of $P(Y) \in GF(2^m)$ defined in (5) for the input operand B (preloaded in register $\langle Y \rangle$). Therefore, we first determine these combinations and after these combinations are computed, we use their results in different computations to optimize the area complexity by reducing the number of signals and consequently number of XOR gates. We propose a method to combine the computations of the P blocks into a Q block as illustrated in Fig. 3b.

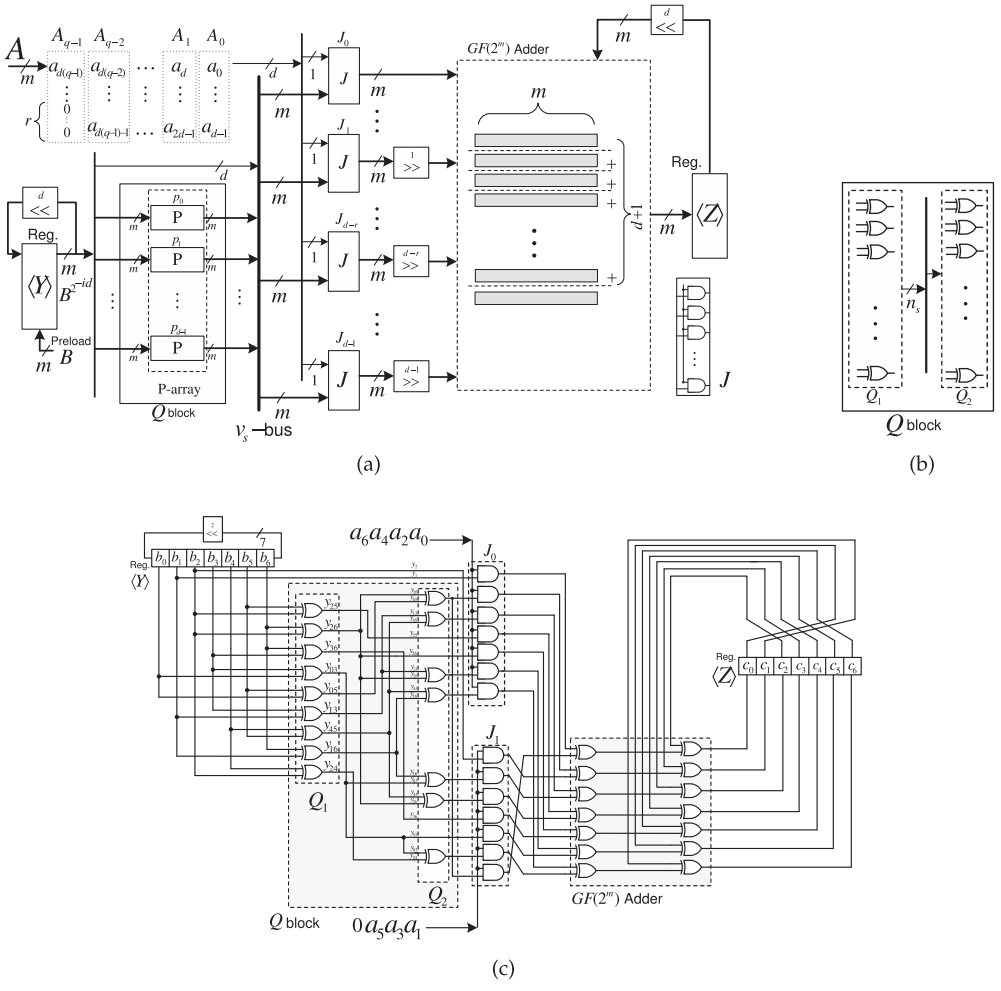


Fig. 3. (a) The proposed architecture for LSD-first DL-SIPO multiplier. (b) An example of the proposed multiplier for type 4 GNB over $GF(2^7)$ with $d = 2$.

The Q block is generated for the digit size d and type T GNB for operand B as $Q(Y) = (P(Y), P(Y) \gg 1, \ldots, P(Y) \gg d-1)$ as illustrated in Fig. 3 where $P(Y) \gg l$, $0 \leq l \leq d-1$ denotes l-fold right cyclic shift of $P(Y) \in GF(2^m)$. As shown in this figure, y_{l+1}, $0 \leq l \leq d-1$ are removed from the block Q as they are correspond to the lines on v_s-bus connected to the register $\langle Y \rangle$. The Q block can also be represented by the $\mathbf{Q}$ matrix as

$$\mathbf{Q} = \begin{pmatrix} \mathbf{R}^{(0)} \\ \mathbf{R}^{(1)} \\ \mathbf{R}^{(2)} \\ \vdots \\ \mathbf{R}^{(l)} \end{pmatrix}_{v_s \times T}, \quad 0 \leq l \leq d-1, \tag{16}$$

where using (9), $\mathbf{R}^{(l)}$ can be obtained by adding the (i,j)th, $1 \leq i \leq m-1$, $1 \leq j \leq T$, entry of the matrix $\mathbf{R} = \mathbf{R}^{(0)}$, i.e., $R(i,j), 0 \leq R(i,j) \leq m-1$ with "$l \bmod m$," as $R(i,j) + l \bmod m$. Also, $v_s = d(m-1) - \frac{d(d-1)}{2}$ is the total number of rows inside the $\mathbf{Q}$ matrix. This is due to the fact that every two $\mathbf{R}^{(i')}$ and $\mathbf{R}^{(i'')}$, $0 \leq i', i'' \leq d-1$, have a common row with the total of $\binom{d}{2} = \frac{d(d-1)}{2}$ in the $\mathbf{Q}$ matrix [16]. Then, as one can see, the multiplication of every bit of A_i in (14) by the outputs of the Q block which is connected to v_s-bus, is performed by J, (J_0 to J_{d-1}) blocks, using (13) where each J block includes m two-input AND gates as shown in Fig. 3a. After the first clock cycle, the content of the register $\langle Y \rangle$ is $B^{2^{-d}}$ and in general it contains $B^{2^{-id}}$ after ith clock cycle. Let $Z(q) \in GF(2^m)$ denotes the field element after the qth clock cycle whose coordinates stored in the m-bit register $\langle Z \rangle$. Then, after one clock cycle, with the use of (14) the register $\langle Z \rangle$ contains

$$C^{(0)} = A_0 B = \sum_{j=0}^{d-1} J^{2^j}(a_j, B^{2^{-j}}). \tag{17}$$

Then, both registers $\langle Y \rangle$ and $\langle Z \rangle$ should be d-fold cyclically shifted to the left to obtain $C^{(1)}, C^{(2)}, \ldots, C^{(q-1)}$, accordingly. The sum of d m-bit intermediate results with one m-bit initial results in the register $\langle Z \rangle$ is performed in the accumulator using the $GF(2^m)$ adder shown in Fig. 3. Therefore, one can verify that considering (15), after qth clock cycle, the register $\langle Z \rangle$ contains

$$Z(q) = \left(\cdots \left(\left(\left(C^{(0)} \right)^{2^{-d}} + C^{(1)} \right)^{2^{-d}} + C^{(2)} \right)^{2^{-d}} + \cdots \right)^{2^{-d}} + C^{(q-1)}. \tag{18}$$

By comparing (15) with (18) one can write $Z(q) = C^{2^{-d(q-1)}} = C^{2^{m+(d-r)}} = C^{2^{d-r}}$. Thus, the coordinates of $C = AB$ can be obtained by $(d-r)$-fold left cyclic shift of the register $\langle Z \rangle$, i.e., $C = (Z(q) \ll d-r)$.

Remark 2. Using the above formulation, one can design similar architecture for the MSD-first digit-level SIPO GNB multiplier.

3.1.3 Complexities

In this section, the complexity of the proposed digit-level SIPO multiplier is given in terms of gate counts and critical-path delay.

The number of rows in the matrix which builds Q is $v_s = d(m-1) - \frac{d(d-1)}{2}$ and each row consists of at most $\frac{T}{2}$ pairs. We divide the Q block into two blocks of Q_1 and Q_2. The block Q_1 contains at most n_s, $n_s \leq v_s \times \frac{T}{2}$, XOR gates with the delay of an XOR gate as shown in Fig. 3b. The block Q_2 consists trees of XOR gates for the GNB, with $T > 2$. The Q_2 block connects its input bus to the v_s-bus having each of its output to be addition of at most T coordinates of $\langle Y \rangle$ which can be obtained by adding at most $\frac{T}{2}$ signals from the output of Q_1. Therefore, if no common subexpression in the Q block are reused, the number of XOR gates in Q_1 and Q_2 of Fig. 3b are at most $v_s\frac{T}{2}$ and $v_s(\frac{T}{2}-1)$, respectively. It is noted that for the case where $d = m$ (i.e., bit-parallel architecture), the upper bound for n_s can be obtained as $\binom{m}{2} = \frac{m(m-1)}{2}$ and hence in general $n_s \leq \min\{\frac{v_sT}{2}, \binom{m}{2}\}$. Also, the number of XOR gates in the $GF(2^m)$ adder (which adds $d+1$ m-bit inputs together) is dm XOR gates. Moreover, the J blocks require dm two-input AND gates. Therefore, based on the above discussions, the following can be stated to obtain the gate count and time complexity of the proposed multiplier architecture.

Proposition 1. *The gate complexities of the proposed LSD-first DL-SIPO multiplier architecture is*

$$\#AND = dm,$$
$$\#XOR \leq v_s(T-1) + dm.$$

Remark 3. The area complexity of proposed LSD-first DL-SIPO multiplier can be further reduced by incorporating a common subexpression elimination algorithm to $n_s + v_s \times (\frac{T}{2}-1) + dm$ XOR gates whose n_s is upper bounded by $n_s \leq \min\{\frac{v_sT}{2}, \binom{m}{2}\}$ and its exact number can be obtained by simulation.

To obtain the maximum clock frequency for the proposed multiplier, one can see that the critical-path delay of the proposed multiplier architecture includes those for the Q_1 and Q_2 blocks (i.e., T_X and $\lceil \log_2 \frac{T}{2} \rceil T_X$, respectively), the J blocks, (i.e., T_A) and the $GF(2^m)$ adder (i.e., $\lceil \log_2(d+1) \rceil T_X$). Then, the total critical-path delay due to delays through the above mentioned blocks is $T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d+1) \rceil)T_X$.

3.1.4 Complexity Reduction

As explained in the previous section, the number of rows inside the $\mathbf{Q}$ matrix is $v_s = d(m-1) - \frac{d(d-1)}{2}$ to generate all signals at the output of $Q(Y)$. As mentioned in Remark 1, the matrix $\mathbf{R}$ contains rows with two equal entries (these entries cancel each other in the formulation). Then, the $\mathbf{Q}$ matrix has some rows with only two entries (i.e., one pair). Based on this fact and the number of times that these pairs are repeated, a subexpression sharing method presented in [25] is used here to obtain the optimized number of pairs in Q_1, i.e., n_s. In the following, we give an illustrative example for the proposed multiplier architecture.

3.2 An Illustrative Example

We consider the multiplication matrix $\mathbf{R}$ for type $T = 4$ GNB over $GF(2^7)$ as follows:

$$\mathbf{R} = \begin{pmatrix} 0 & 2 & 5 & 6 \\ 1 & 3 & 4 & 5 \\ 2 & 5 & \underline{3} & \underline{3} \\ 2 & 6 & \underline{0} & \underline{0} \\ 1 & 2 & 3 & 6 \\ 1 & 4 & 5 & 6 \end{pmatrix}_{(6\times 4)}. \tag{19}$$

This matrix can be obtained from the location of nonzero entries (excluding the first row) of the multiplication matrix $\mathbf{M}$ as

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}_{7\times 7}.$$

Having the digit size to be $d = 2$, the matrix $\mathbf{Q}_{(11\times 4)}$ can be generated as

$$\mathbf{Q} = \begin{pmatrix} \mathbf{R}^{(0)} \\ \mathbf{R}^{(1)} \end{pmatrix} = \begin{pmatrix} \boxed{0\ 2\ 5\ 6} \\ 1\ 3\ 4\ 5 \\ 2\ 5\ \underline{3}\ \underline{3} \\ 2\ 6\ \underline{0}\ \underline{0} \\ 1\ 2\ 3\ 6 \\ 1\ 4\ 5\ 6 \\ 1\ 3\ 6\ 0 \\ 2\ 4\ 5\ 6 \\ 3\ 6\ \underline{4}\ \underline{4} \\ 3\ 0\ \underline{1}\ \underline{1} \\ 2\ 3\ 4\ 0 \\ \boxed{2\ 5\ 6\ 0} \end{pmatrix}_{11\times 4} \begin{matrix} \} \mathbf{R}^{(0)} \\ \\ \\ \\ \\ \\ \} \mathbf{R}^{(1)} \\ \\ \\ \\ \end{matrix} \quad \text{Removed} \longrightarrow$$

In this matrix, $\mathbf{R}^{(1)}$ is obtained by adding the (i,j)th entry of $\mathbf{R} = \mathbf{R}^{(0)}$ by "1 mod 7." As one can see, the number of rows in this matrix is $v_s = 2 \times (7-1) - \binom{2}{2} = 11$ (as $\mathbf{R}^{(0)}$ and $\mathbf{R}^{(1)}$ have a common row which is removed from this matrix) and it has $2d = 4$ rows with just two entries (as the equal underlined entries cancel each other in those four rows). Then, we first collect these pairs (in rows with two entries), i.e., (2, 5), (2, 6), (3, 6), and (3, 0) as a pairset to initialize $\mathbf{Q}_1$ matrix. The numbers of times that these pairs are repeated are 2, 3, 2, and 2, respectively. Then, applying the common subexpression elimination algorithm presented in [25], one can obtain the pairs inside the matrix $\mathbf{Q}_1$ as $\mathbf{Q}_1 = \{y_{25}, y_{26}, y_{36}, y_{30}, y_{05}, y_{13}, y_{45}, y_{16}, y_{24}\}$, where $y_{ij} = y_i + y_j$ and $n_s = 9$ are the number of pairs in Q_1. Also, as each row in $\mathbf{Q}$ needs $(\frac{T}{2}-1)$ gates excluding the rows with only two entries (which is $2d$ here) and there are v_s rows in total, then $v_s(\frac{T}{2}-1) - 2d = 7$ XOR gates in block Q_2 is required to produce the outputs of $Q(Y)$. The architecture of the proposed multiplier over $GF(2^7)$ for $d = 2$ is depicted in Fig. 3c. Therefore, the complexity of the

TABLE 1
Contents of Variables in the Proposed Architecture for LSD-First DL-SIPO Type 4 GNB Multiplier over $GF(2^7)$

Clock	LSD-First			
j	A	Y	Acc	Z
0	–	$B = 1100011$	–	0000000
1	11	1100011	0111010	0111010
2	00	0001111	0000000	1101001
3	01	0111100	1100111	1000000
4	10	1110001	1111010	$C^2 = 1111000$

presented improved DL-SIPO multiplier is $n_s + v_s(\frac{T}{2} - 1) - 2d + dm = 30$ XOR gates. Note that the unoptimized structure (without common subexpression sharing) requires $(d(m-1) - \frac{d(d-1)}{2})(T-1) - 2d + dm = 43$ XOR gates and the architecture proposed in [28] requires $m(dT+1) - d = 61$ XOR gates. Also, the critical-path delay is $T_A + 4T_X$.

For the multiplier operation, as one can see in Fig. 3c, operand A is grouped into four digits as $A_0 = (a_0, a_1)$, $A_1 = (a_2, a_3)$, $A_2 = (a_4, a_5)$, and $A_3 = (a_6, 0)$, each with the size of 2 bits, i.e., $d = 2$. Before starting the clock, the register $\langle Y \rangle$ is loaded with the coordinates of $B = (b_0, b_1, \ldots, b_6)$ and the register $\langle Z \rangle$ is cleared to zero, i.e., $\langle Z \rangle = (0, 0, \ldots, 0)$. Then, in the first clock cycle, two LSD bits, i.e., a_0 and a_1 of operand A, are the inputs of the corresponding AND gates. One can realize that after $q = \lceil \frac{7}{2} \rceil = 4$ clock cycles, the result of $C^{2^{d-r}} = C^2$ is available in parallel at the register $\langle Z \rangle$. The contents of registers are given in Table 1 for $A = B =$ (11000011). Note that as mentioned before, the result of multiplication $C = AB$ is obtained after one $(d - r = 1)$ left cyclic shift of the content of register $\langle Z \rangle$ at the last clock cycle, i.e., $C = (Z(q) \ll 1) = 1110001$.

3.2.1 Simulations

In an effort to obtain the exact complexity of the improved multiplier, a Matlab code is written to generate common pairs and signals used in the blocks Q_1 and Q_2 of the proposed architectures in Fig. 3a. As shown in Figs. 4a, 4b, and 4c, we first plot the upper bound of the number of required XOR gates (i.e., for the architectures without applying common subexpression sharing) given in Proposition 1 versus the digit-size for the three fields $GF(2^{163})$ $(T = 4)$, $GF(2^{283})$ $(T = 6)$, and $GF(2^{233})$ $(T = 2)$, recommended by NIST for ECDSA [7] and compare to the original architectures. Then, for $T = 4$ over $GF(2^{163})$ and $T = 6$ over $GF(2^{283})$ the exact number of XOR gates after applying common subexpression sharing (applicable only for $T > 2$) are obtained and plotted for the improved DL-SIPO GNB multiplier in Figs. 4a and 4c, respectively. For a given number of clock cycle, q, $1 \le q \le m$, the least value of digit sizes in the form of $d = \lceil \frac{m}{q} \rceil$, $1 \le d \le \lceil \frac{m}{2} \rceil$, is incorporated so that the area complexity is optimized. From Figs. 4a and 4c, one can see that as the digit size increases, more common pairs are found. For the digit size $d = m = 163$, the total number of XOR gates required in the original DL-SIPO multiplier is 66,178 gates, whereas, the modified one requires 50,401 XOR gates for $GF(2^{163})$. This means that the complexity of the proposed improved DL-SIPO multiplier architecture is about 24 percent less than the original multiplier for the bit-parallel structure. In Figs. 5a, 5b, and 5c we plot the delay (in terms of number of cycles) versus

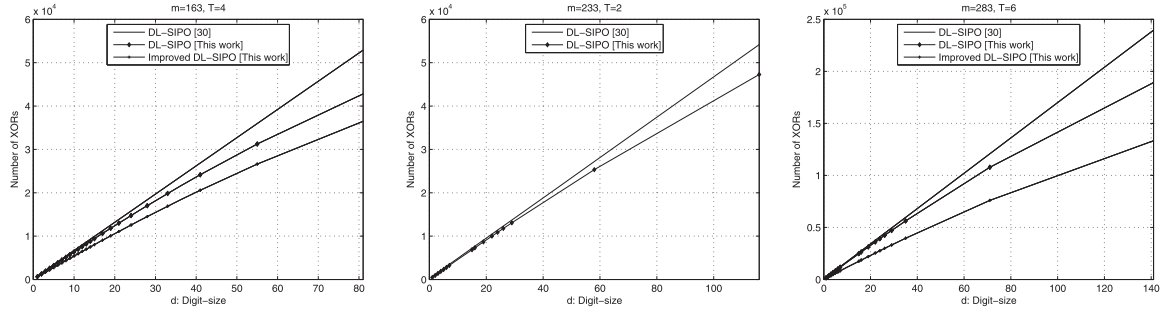


Fig. 4. Comparison among the numbers of XOR gates required in [28], the new architecture, and improved digit-level SIPO multiplier architectures for (a) type $T = 4$ GNB over $GF(2^{163})$ and (b) type $T = 2$ GNB over $GF(2^{233})$ and (c) type $T = 6$ GNB over $GF(2^{283})$.

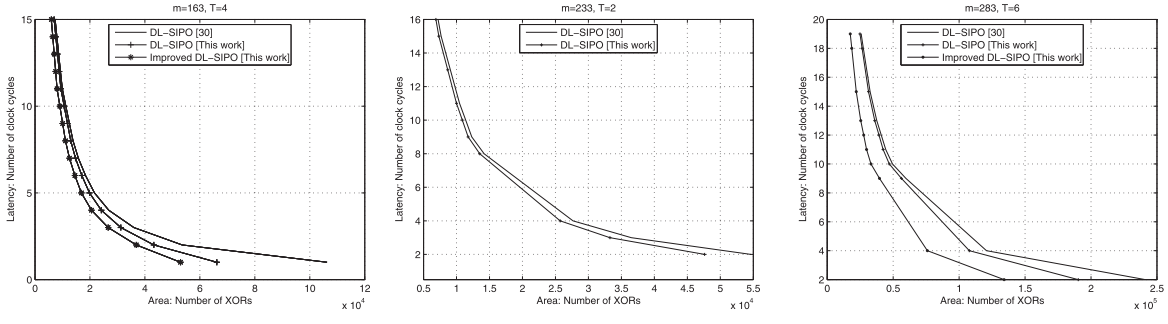
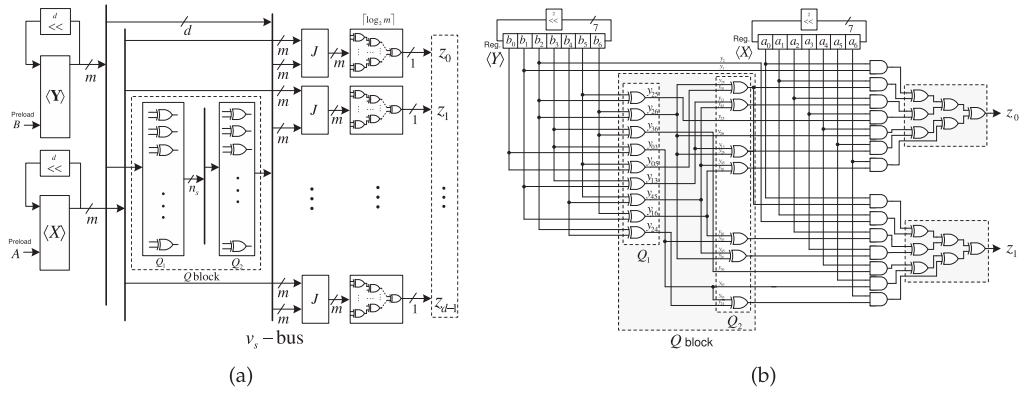
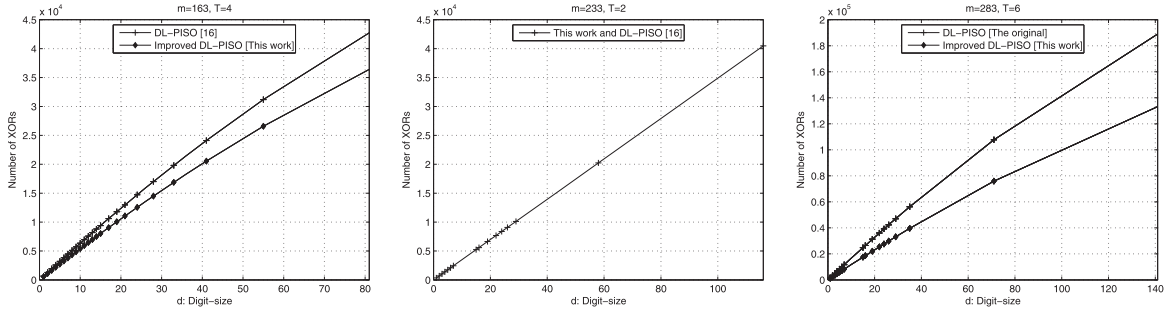


Fig. 5. Comparison of the latency (number of clock cycles) in terms of area (number of XOR gates) for the architecture of [28], the new architecture, and improved digit-level SIPO multiplier architectures for (a) type $T = 4$ GNB over $GF(2^{163})$, (b) type $T = 2$ GNB over $GF(2^{233})$ and (c) type $T = 6$ GNB over $GF(2^{283})$.



(a) (b)

Fig. 6. (a) The architecture of the improved digit-level PISO GNB multiplier architecture with the LSD-first output. (b) The improved architecture of type 4 GNB multiplier over $GF(2^7)$ with $d = 2$.



(a) (b) (c)

Fig. 7. Comparison among the numbers of XOR gates required in the original [16] and improved digit-level PISO multiplier architectures for (a) type $T = 4$ GNB over $GF(2^{163})$, (b) type $T = 2$ GNB over $GF(2^{233})$, and (c) type $T = 6$ GNB over $GF(2^{283})$.

the area (in terms of number of XOR gates) for the fields $GF(2^{163})(T = 4)$, $GF(2^{233})(T = 2)$, and $GF(2^{283})(T = 6)$.

3.3 Low-Complexity Digit-Level PISO GNB Multiplier

In this section, we present a low-complexity architecture for the digit-level PISO GNB multiplier [16] presented in Section 2.3. The improvement of the new architecture is based on a formulation of the multiplication operation, which is given in the following.

3.3.1 Improved Architecture

In this section, similar to the previous section, we present an improved architecture for DL-PISO GNB multiplier and reduce its area complexity. As shown in Fig. 2, the digit-level PISO multiplier architecture has several P blocks that use the same combination of the input operand B (preloaded in the register $\langle Y \rangle$). We combine the computations of the parallel computed functions into a Q block (which is the same as the one presented in previous section for DL-SIPO architecture) as illustrated in the architecture in Fig. 6. As shown in this figure, y_{1+d}s are removed from the block Q as they correspond to the lines on v_s-bus connected to the register $\langle Y \rangle$. The v_s-bus contains all signals to generate all different terms required in (7). These signals are implemented by the blocks of Q_1 and Q_2 inside the Q block. We first use the block Q_1 to implement all pairs required for all signals in (7). In this architecture, each J block consists of m 2-input AND gates to implement (8). Then, a level of XOR trees are utilized to implement all $z_0, z_1, \ldots, z_{d-1}$ coordinates in (8). The proposed improved architecture provides the LSD of multiplication at the first clock cycle (LSD-first).

For the purpose of illustration, the improved architecture of DL-PISO ($d = 2$) for type 4 GNB over $GF(2^7)$ is shown in Fig. 6b. As shown in this figure, the Q_1 and Q_2 blocks are generated for the given matrix $\mathbf{R}$ in (19). The registers $\langle X \rangle$ and $\langle Y \rangle$ should be initialized with the coordinates of A and B and then after each clock cycle 2 bits of $C = AB$ become available at the output.

In the following, we derive the complexity of the improved LSD-first DL-PISO GNB multiplier.

3.3.2 Complexities

To determine the area and time complexities of the presented architecture, the following is stated.

Proposition 2. *For type T GNB over $GF(2^m)$, the improved digit-level PISO GNB multiplier requires dm AND gates and $n_s + v_s \times (\frac{T}{2} - 1) + d(m-1)$ XOR gates. Also, the critical-path delay of the improved architecture is the same as the original structure, i.e., $T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$.*

Proof. The proof is similar to the one presented in Section 3.1.3. □

We further optimize the number of XOR gates required for the improved LSD-first DL-PISO GNB multiplier incorporating a complexity reduction algorithm proposed in [25]. The results of simulations obtained for different digit-size are plotted in Figs. 7a, 7b, and 7c, for $m = 163$, $m = 233$, and $m = 283$, respectively. It is noted that our proposed improved architecture requires fewer number of XOR gates for type $T > 2$. However, it provides no reduction in the number of XOR gates for type $T = 2$ in comparison to the original architecture as shown in Fig. 7b.

TABLE 2
Comparison of Digit-Level Type T GNB Multipliers over $GF(2^m)$

Multiplier Architecture	# AND gates	# XOR [1] gates	# Reg.	Critical-Path delay	Output	Input	
						A	B
DL-PISO [8]	$d(Tm-T+1)$	$2v_p.T$	$2m$	$T_A+(\lceil\log_2(Tm-T+1)\rceil)T_X$	Serial	Parallel	Parallel
DL-PISO [22]	dm	$2v_p.T$	$2m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$	Serial	Parallel	Parallel
DL-PIPO [17]	dm	$2v_p.T+d$	$3m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2(d+1)\rceil)T_X$	Parallel	Parallel	Parallel
DL-PIPO [16]	dm	$\leq v_p.T+\frac{d}{2}(m+1)$	$3m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2(d+1)\rceil)T_X$	Parallel	Parallel	Parallel
DL-PIPO [25]	dm	$\leq v_p(T-1)+dm$	$3m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2(d+1)\rceil)T_X$	Parallel	Parallel	Parallel
DL-SIPO [28]	dm	$2v_p.T+d(T-1)+m$	$2m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 d\rceil+1)T_X$	Parallel	Serial	Parallel
DL-SIPO (Fig. 3)[2]	dm	$\leq v_s(T-1)+dm$	$2m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2(d+1)\rceil)T_X$	Parallel	Serial	Parallel

1. $v_p=\frac{d(m-1)}{2}$ and $v_s=d(m-1)-\frac{d(d-1)}{2}$.
2. Without applying common subexpression elimination algorithm.

TABLE 3
Area and Time Complexity Comparison of Bit-Parallel Type T GNB Multiplier Architectures over $GF(2^m)$

Multiplier	# AND	# XOR	Critical-Path delay
Type $T=2$			
Bit-parallel [19]	m^2	$1.5m(m-1)$	$T_A+(1+\lceil\log_2 m\rceil)T_X$
Bit-parallel [29]	m^2	$1.5m(m-1)$	$T_A+(1+\lceil\log_2 m\rceil)T_X$
Type $T\geq 2$ (T is even)			
Bit-parallel [22]	m^2	$Tm(m-1)$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$
Bit-parallel [21]	m^2	$\leq\frac{T+1}{2}m(m-1)$	$T_A+(\lceil\log_2(Tm-T+1)\rceil)T_X$
DL-PISO ($d=m$) [16]	m^2	$\leq\frac{T+1}{2}m(m-1)$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$
DL-PIPO ($d=m$) [17]	m^2	$\leq Tm(m-1)+m$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$
DL-PIPO ($d=m$) [25]	m^2	$\leq\frac{T+4}{4}m(m-1)$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$
DL-SIPO ($d=m$) [28]	m^2	$\leq(T-1)m^2+m(m-1)$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$
This work [Remark 3 ($d=m$)][1]	m^2	$\leq\frac{T+4}{4}m(m-1)$	$T_A+(\lceil\log_2 T\rceil+\lceil\log_2 m\rceil)T_X$

1. Without applying common subexpression elimination algorithm.

3.4 Complexity Comparison

In Table 2, the time and area complexities of the presented DL-SIPO multiplier (before applying common subexpression elimination algorithm) are compared with the ones, namely, DL-SIPO [28], DL-PISO [16], and DL-PIPO [17] multipliers as they appear to be the most recently proposed works as well as the old ones available in the literature. It is noted that our presented multiplier architecture (Fig. 3) requires fewer number of gates than the previously proposed ones DL-SIPO [28] and DL-PIPO [17]. Also, as seen in this table, in terms of time complexity our presented multiplier (Fig. 3) is favorably comparable with the DL-SIPO [28]. Moreover, in Fig. 4, the area complexity of the improved architecture over $GF(2^{163})$ and $GF(2^{283})$ after applying the common subexpression elimination algorithm [25] is illustrated in terms of different digit sizes and compared with the ones of its counterpart [28]. As illustrated in Figs. 4 and 7, the presented improved architectures require fewer XOR gates than the one proposed in [28] and the original one proposed in [16], respectively.

It is interesting to note that the ρ matrix presented in [25] is for the DL-PIPO architecture and its size is $\frac{d(m-1)}{2}\times T$. Also, its both input operands are available and the results are obtained in parallel after $q=\lceil\frac{m}{d}\rceil$ clock cycles. This results in reducing the size of ρ matrix. However, in this work, for DL-SIPO and DL-PISO architectures, the size of matrix $\mathbf{Q}$ is $(d(m-1)-\binom{d}{2})\times T$. Note that in the DL-SIPO multiplier, only one operand is fully available and for the DL-PISO multiplier, every digit of the output should be obtained in every clock cycle. Therefore, the multiplication matrix is not the same as the one proposed for ρ matrix in [25]. Furthermore, the pairs (XOR gates) that build the blocks ρ and Q are different for each digit size. It is worth mentioning that the proposed DL-SIPO multiplier architectures can be easily scaled up to the bit-parallel type. In Table 3, the area and time complexity of bit-parallel version of our presented multiplier is compared with the counterparts. As one can see, our architecture requires at most $\frac{T+4}{4}m(m-1)$ XOR gates (without employing complexity reduction algorithm) which is the smallest one in terms of number of required XOR gates having the same critical-path delay.

In the following section, we propose a new hybrid multiplier which is composed of the DL-PISO and DL-SIPO multiplier architectures presented in this section.

4 A New Hybrid Structure for Double Multiplication

4.1 Hybrid Multiplication

The discussion of the previous section is dealt with low-complexity and improved DL-SIPO and DL-PISO GNB multipliers. Based on the information provided there, we here present a new hybrid structure by connecting the output of the DL-PISO multiplier to the serial input of the DL-SIPO multiplier and build a new hybrid multiplier. This entire hybrid multiplier performs two multiplications simultaneously, where the results are available in parallel after $\lceil\frac{m}{d}\rceil+1$ clock cycles assuming that one clock cycle is required to load the output of the first multiplier (stored in the register) to the input of the second multiplier. The structure of the proposed hybrid multiplier is illustrated in Fig. 8a. It computes $E=A\times B\times D$ over $GF(2^m)$.

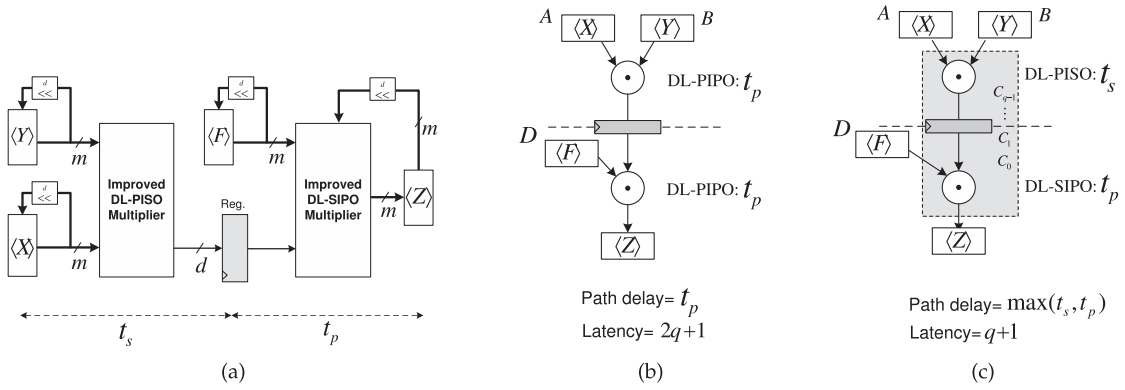


Fig. 8.(a) Proposed structure for the hybrid multiplier. (b) Two digit-level multipliers with parallel output operating in two separate steps. (c) A hybrid multiplier operating in one step using the proposed DL-PISO and DL-SIPO multipliers.

4.1.1 Traditional Multiplication Scheme

In Fig. 8b, two digit-level multipliers with parallel output (DL-PIPO) are employed to compute $E = A \times B \times D$, $E \in GF(2^m)$. Let us assume that registers $\langle X \rangle$, $\langle Y \rangle$, and $\langle F \rangle$ are preloaded with the operands A, B, and D, respectively. Also, the register $\langle Z \rangle$ should be initialized with $0 \in GF(2^m)$. The top multiplier (of Fig. 8b) requires q clock cycle to compute $C = A \times B$ and store the results to the m-bit register. Also, the bottom multiplier requires q clock cycles to perform $(AB) \times D$ and store it to the register $\langle Z \rangle$. Therefore, to obtain the results in register $\langle Z \rangle$, $2q + 1$ clock cycles are required. It should be noted that the critical-path delay is equal to t_p which is the delay of a digit-level GNB multiplier with parallel output. Then, the required time to compute E is $T = t_p \times (2q + 1)$.

4.1.2 Hybrid Multiplication Scheme

Now, we consider Fig. 8c, which depicts the use of a hybrid multiplier which is composed of a digit-level PISO GNB multiplier and a LSD-first digit-level SIPO multiplier. This multiplier performs two dependent multiplications to reduce the latency to the one of one multiplication. Let us assume that $C \in GF(2^m)$ be the product of A and B, i.e., $C = AB$. Based on the output of digit-level PISO multiplier, C will be available from its LSD as $C_0, C_1, \ldots, C_{q-1}$ in each clock cycle. In the first clock cycle it provides the first digit of C, i.e., $C_0 = (c_0, c_1, \ldots, c_{d-1})$. In the second clock cycle, the bottom multiplier (i.e., DL-SIPO) multiplies the first digit of C, i.e., C_0 by D (stored in register $\langle F \rangle$) and the top multiplier computes the second digit of C, i.e., $C_1 = (c_d, c_{d+1}, \ldots, c_{2d-1})$. Then, one can realize that after $q + 1$ clock cycles, register $\langle Z \rangle$ contains the result of multiplication of $E = A \times B \times D$. The critical-path delay of the hybrid multiplier is equal to the maximum of the delays for the DL-PISO and DL-SIPO multipliers, i.e., $t_s = \max\{t_p, t_s\}$, and consequently one can obtain the time of multiplication as $T = t_s \times (q + 1)$.

Based on the information provided above, one can state the following to obtain the area and time complexities of the presented hybrid multiplier.

Proposition 3. *The proposed hybrid multiplier architecture requires* $\leq 2v_s(T-1) + 2dm - d$ *XOR gates,* $2dm$ *AND gates, four* m*-bit registers, and one* d*-bit register. Also, its critical-path delay is equal to* $T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil)T_X$ *which is due to the delays through logic gates in the path with longer critical-path delay (i.e., DL-PISO architecture).*

4.1.3 Analysis

In Table 4, the latency and time delay of the proposed hybrid multiplier is investigated in terms of different digit sizes for type 4 GNB over $GF(2^{163})$. As shown in this table, the latency, critical-path delay (CPD), and time to perform the entire multiplication are given for different digit sizes d, $7 < d < 128$. For the traditional method, i.e., the structure of Fig. 8b, the latency is $2q + 1$ while for the hybrid structure, i.e., Fig. 8c, the latency is $q + 1$. The time of multiplication for the proposed hybrid structure is $T = (q + 1)T_A + (10q + 10)T_X$ which is about 17 percent less than the general method for smaller digit sizes, e.g., $7 < d \leq 15$ and is 38 percent less while choosing larger digit sizes, e.g., $31 < d \leq 63$. Therefore, the proposed hybrid structure in Fig. 8c reduces the latency and consequently the total time of multiplication and outperforms the one depicted in Fig. 8b.

The proposed hybrid architecture is particularly applicable for reducing the latency whenever there are repeated multiplications with data dependency.

TABLE 4
Time Delay Evaluation of the Proposed Structure for Type 4 GNB over $GF(2^{163})$

digit-size	Structure of Fig. 8b			Hybrid Structure in Fig. 8c		
	Latency	CPD	Time	Latency	CPD	Time
$7 < d \leq 15$	$2q+1$	$T_A + 6T_X$	$(2q+1)T_A + (12q+6)T_X$	$q+1$	$T_A + 10T_X$	$(q+1)\times$ $(T_A + 10T_X)$
$15 < d \leq 31$		$T_A + 7T_X$	$(2q+1)T_A + (14q+7)T_X$			
$31 < d \leq 63$		$T_A + 8T_X$	$(2q+1)T_A + (16q+8)T_X$			
$63 < d \leq 127$		$T_A + 9T_X$	$(2q+1)T_A + (18q+9)T_X$			

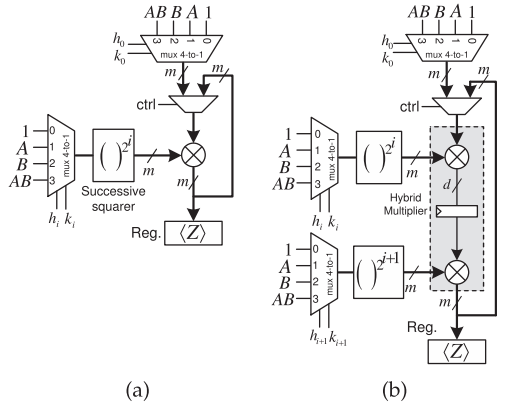


Fig. 9. Architectures for multiplexer-based double-exponentiation. (a) With one multiplier. (b) With incorporating the proposed hybrid multiplier.

4.2 Applications of the Proposed Hybrid Multiplier

In this section, we provide some of the applications of the proposed hybrid multiplier architecture whenever high-speed double-multiplications are required.

4.2.1 Double-Exponentiation

The exponentiation on an Abelian group (e.g., finite fields) is one of the most important arithmetic operations for public key cryptography such as Diffie-Hellman [4] key agreement, RSA, and encoding the Reed Solomon codes [30], [31], and [32]. The exponentiation is usually accomplished by performing repeated field multiplications and squarings [30]. Let A and$B$ be two field elements and K and H be two integers. Then, the computation of $A^K B^H$ (denoted by double-exponentiation) is a crucial operation for cryptographic applications such as Schnorr- and ElGamal-like signature verifications [26] and [3]. Computing double-exponentiation is presented in [30] by multiplying the result of single exponentiations. Such an scheme is not the most efficient method and efficient computation of double-exponentiation is required [32].

As explained before, under normal basis representation of field elements squarings are free. Thus, to speed up double-exponentiation one requires to reduce the total number of field multiplications as well as the complexity of each multiplication. The former reduces the latency (in terms of number of clock cycles) while the latter improves the execution time of a multiplier (in terms of propagation delay through logic gates). Based on the discussion regarding low-complexity multipliers presented in the previous sections, we reduce the latency of double-exponentiation using the proposed hybrid multiplier architecture. The following is used in [31] to compute the double exponentiation.

Lemma 3 ([31]). *Let A and B be two field elements on $GF(2^m)$ and represented by normal basis and assume K and H be the two positive integers represented by $K = (k_{m-1}, \ldots, k_1, k_0)_2$ and $H = (h_{m-1}, \ldots, h_1, h_0)_2$, respectively. Double-exponentiation of the form $A^K B^H$ is computed by*

$$\begin{aligned} A^K B^H &= A^{k_0 + k_1 2 + \cdots + k_{m-1} 2^{m-1}} B^{h_0 + h_1 2 + \cdots + h_{m-1} 2^{m-1}} \\ &= (A^{k_0} B^{h_0})(A^{k_1} B^{h_1})^2 \cdots (A^{k_{m-1}} B^{h_{m-1}})^{2^{m-1}} \\ &= (\ldots (A^{k_{m-1}} B^{h_{m-1}})^2 A^{k_{m-2}} B^{h_{m-2}})^2 \ldots)^2 A^{k_0} B^{h_0}. \end{aligned}$$

The architecture of a multiplexer-based double-exponentiation using one multiplier is given in Fig. 9a. It is assumed in [31] that AB is precomputed. As seen in this figure, the result of double-exponentiation is available after $m-1$ iterations, i.e., $(m-1) \times q$, $q = \lceil \frac{m}{d} \rceil$ clock cycles. In Fig. 9b, we have proposed a new architecture by employing our proposed hybrid multiplier architecture. This hybrid multiplier performs two multiplications with the latency of one multiplication and as seen the double-exponentiation results will be in the register $\langle Z \rangle$ available after $\lceil \frac{m-1}{2} \rceil$ iterations, i.e., $\lceil \frac{m-1}{2} \rceil \times (q+1)$ clock cycles. This is due to the fact that in each iteration 2 bits of K, $k_i k_{i+1}$ and H, $h_i h_{i+1}$ are processed from their LSB in parallel. One should note that as the representation of field elements are under normal basis, thus computation of repeated squarings is free. Therefore, our proposed scheme reduces the latency of the double-exponentiation based on choosing efficient values for digit-size d. It is noted that the fast operation is achieved at the expense of extra area. More importantly, one can obtain a tradeoff between time and area by choosing suitable values for d. The presented architectures for double-exponentiation can be easily modified to eliminate the multiplication by 1, i.e., $(1, \ldots, 1, 1)$ in normal basis, whenever h_i and k_i are both zero. However, for the sake of simplicity we do not investigate it here.

In Table 5, the proposed architecture for double-exponentiation is compared to the counterparts. As one can see from this table, our scheme employing the hybrid multiplier has the smallest latency at the cost of increasing the area. As a result, one can obtain a tradeoff between the latency and area with choosing the digit-size d.

In [32], a new exponentiation algorithm based on split exponents is proposed. Using normal basis representation and the proposed hybrid multiplier architecture, it can be improved.

4.2.2 Attacking ECC2K-130

In [33], Fan et al. have performed an extensive investigation to solve one of the Certicom elliptic curve discrete logarithm problem (ECDLP) challenges, ECC2K-130 using Pollard's rho method [34]. They have focused on Koblitz curves over

TABLE 5
Comparison of the Proposed Double-Exponentiation Scheme over $GF(2^m)$ with the Counterparts

Scheme	Basis	Latency (# of Multiplications)	Area (# of Multipliers)	Type of Multiplier	# XORs	# AND
[30]	NB	$2m+1$	1	MO	$dT(m-1)$	dm
[30]	NB	$m+1$	2	MO	$2dT(m-1)$	$2dm$
[31]	NB	m	1	MO	$dT(m-1)$	dm
This work[1]	GNB	$\lceil \frac{m-1}{2} \rceil$	1	Hybrid	$\leq (2d(m-1) - d(d-1))(T-1) + 2dm - d$	$2dm$

1. Note that d should be chosen smaller in comparison to the counterparts.

TABLE 6
FPGA Implementation Results for the Presented Multiplier Architectures for Type 4 GNB over $GF(2^{163})$ for Different Digit Sizes Using Xilinx Virtex-4 xc4vlx100-ff1148 Device

digit size	clock cycle $q = \lceil \frac{m}{d} \rceil$	Improved DL-SIPO (Fig. 3)					Improved DL-PISO (Fig. 6)				
		Slices	FF	LUT	CPD [ns]	Time [ns]	Slices	FF	LUT	CPD [ns]	Time [ns]
11	15	1,691	326	3,365	4.8	72.0	1,899	444	3,912	5.7	85.5
21	8	3,099	326	6,185	5.8	46.4	3,754	408	6,995	6.1	48.8
33	5	5,739	326	10,281	6.3	31.5	5,908	365	10,735	6.8	34.0
41	4	7,229	326	12,783	6.5	26.0	7,385	378	13,218	6.9	27.6
55	3	9,323	326	16,715	6.7	20.1	9,678	419	17,348	7.3	21.9

$GF(2^{131})$ and because of performing several squarings, normal basis is incorporated [33]. Each iteration of their method requires five multiplications that cannot be reduced by employing parallel multipliers due to data dependencies. However, our proposed hybrid multiplier for GNB (for type 2) can be incorporated to reduce the latency of each iteration to four multiplications and improve the overall speed of the attack.

It is worth mentioning that the proposed hybrid multiplier architecture can be used in other applications. For example, after small modifications it could be employed to reduce the latency of computing multiplicative inversion using Itoh-Tsuji's method [35] (based on Fermat's Little Theorem). Also, it can be used to reduce the latency of point multiplication over binary elliptic curves for ECC-based cryptography.

5 FPGA and ASIC Implementations

In this section, we implement the presented architectures in the previous sections to evaluate their area and time requirements. We have selected the Xilinx Virtex-4 xc4vlx100-ff1148 device as the target FPGA. The proposed multiplier architectures are modeled in VHDL and synthesized for different digit sizes using XST of Xilinx ISE version 12.1 design software. Also, 65-nm Complementary Metal-Oxide-Semiconductor (CMOS) library has been chosen for the synthesis on application-specific integrated circuit (ASIC) technology. The proposed architectures synthesized using Synopsys Design Vision which is a GUI for Synopsys Design Compiler tools. The correctness of the multiplier architectures is verified by Xilinx ISE Simulator (ISim). For the FPGA implementations, the optimization goal is set to the speed (i.e., default) and optimization effort is set to normal and the area (Slices, LUTs, and FFs) and timing (ns) for the critical-path delays (CPD) are obtained for different digit sizes. It is noted that the results of the implementations on FPGA, are all after post place and route results. For the ASIC implementations, the map effort is set to medium with a target clock period of 5 ns and the area (μm^2) and timing (ns) are obtained for each of the designs. We have implemented the proposed architectures for digit-level PISO and LSD-first SIPO multipliers for different digit sizes on FPGA and synthesized for ASIC. The results of the implementations for different digit sizes are reported in Tables 6 and 7 for FPGA and ASIC, respectively. As one can see, the total time of multiplication is computed by multiplying the number of clock cycles q, by the critical-path delay. Also, the proposed hybrid multiplier architecture is implemented and the area and timing results are reported in Table 8. The total time of double-multiplication (i.e., multiplying three field elements together) is calculated by multiplying the number of clock cycles $q+1$, by the critical-path delay for the different digit sizes. We note that one can reduce the critical-path delay of the proposed hybrid architecture by pipelining the multiplier architectures and maintain high-throughput performance. It should be noted that for any particular application the digit-size should be chosen in such a way to achieve highest performance considering the time-area tradeoffs.

6 Conclusions

In this paper, we have presented a low-complexity digit-level SIPO GNB multiplier and an improved digit-level PISO multiplier architecture over $GF(2^m)$. Then, we have proposed a new hybrid architecture by connecting the output of the digit-level PISO multiplier to the input of the digit-level SIPO multiplier. The proposed hybrid multiplier architecture performs double-multiplication with the same number of clock cycles required as the one for one multiplication. The proposed hybrid multiplier has been employed to reduce the latency of double-exponentiation. We have evaluated the performance of the proposed hybrid

TABLE 7
ASIC Synthesis Results for the Presented Improved Multiplier Architectures for Type 4 GNB over $GF(2^{163})$ for Different Digit Sizes Using 65-nm CMOS Standard Technology

digit size	Clock cycle $q = \lceil \frac{m}{d} \rceil$	Improved DL-SIPO (Fig. 3)			Improved DL-PISO (Fig. 6)		
		Area [μm^2]	CPD [ns]	Time [ns]	Area [μm^2]	CPD [ns]	Time [ns]
11	15	34,278.4	0.93	13.95	34,837.4	1.38	20.70
21	8	63,283	1.56	12.48	63,397.2	1.85	14.8
33	5	97,420.4	2.16	10.80	97,804.2	2.37	11.85
41	4	120,295	2.57	10.28	121,356	2.74	10.96
55	3	160,298.3	3.25	9.75	161,494.8	3.55	10.65

TABLE 8
ASIC and FPGA Implementation Results for the Proposed Low-Complexity Hybrid Multiplier Architecture (Fig. 8) over $GF(2^{163})$ for Different Digit Sizes

digit size	Latency $q+1$	ASIC			FPGA				
		Area $[\mu m^2]$	CPD $[ns]$	Time $[ns]$	Slice	FF	LUT	CPD $[ns]$	Time $[ns]$
11	16	69,048	1.38	22.08	3753	677	7321	5.7	91.2
21	9	127,053	1.85	16.65	6753	705	13,258	6.5	61.2
33	6	195,170	2.37	14.22	11,306	811	21,023	6.8	40.8
41	5	242,096	2.87	14.35	14,589	724	25,928	6.9	34.5
55	4	321,692	3.65	14.60	19,030	763	34,118	7.4	29.6

architecture for different digit sizes and for practical purposes it has been implemented in FPGA and ASIC and the area and timing results have been presented.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive comments. This work has been supported in part by a Natural Sciences and Engineering Council (NSERC) discovery grant awarded to Arash Reyhani-Masoleh. The authors would like to thank Canadian Microelectronics Corporation (CMC) Microsystems for providing the required infrastructure and CAD tools that have been used in this work.

REFERENCES

[1] V.S. Miller, "Use of Elliptic Curves in Cryptography," *Proc. Advances in Cryptology (Crypto),* pp. 417-426, 1986.
[2] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. of Computation,* vol. 48, pp. 203-209, 1987.
[3] T.E. Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory,* vol. 31, no. 4, pp. 469-472, July 1985.
[4] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory,* vol. 22, no. 6, pp. 644-654, Nov. 1976.
[5] D.W. Ash, I.F. Blake, and S.A. Vanstone, "Low Complexity Normal Bases," *Discrete Applied Math.,* vol. 25, no. 3, pp. 191-210, 1989.
[6] IEEE Std 1363-2000, "IEEE Standard Specifications for Public-Key Cryptography," Jan. 2000.
[7] US Dept. of Commerce/NIST, "National Institute of Standards and Technology," *Digital Signature Standard, FIPS Publications 186-2,* Jan. 2000.
[8] J. Massey and J. Omura, *Computational Method and Apparatus for Finite Arithmetic,* US Patent 4587627, Washington, D.C., 1986.
[9] G. Feng, "A VLSI Architecture for Fast Inversion in $GF(2^m)$," *IEEE Trans. Computers,* vol. 38, no. 10, pp. 1383-1386, Oct. 1989.
[10] T. Beth and D. Gollman, "Algorithm Engineering For Public Key Algorithms," *IEEE J. Selected Areas in Communications,* vol. 7, no. 4, pp. 458-466, May 1989.
[11] 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 Integration (VLSI) Systems,* vol. 18, no. 8, pp. 1234-1238, Aug. 2010.
[12] W. Geiselmann and D. Gollmann, "Symmetry and Duality in Normal Nasis Multiplication," *Proc. Sixth Symp. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC),* pp. 230-238, July 1989.
[13] 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.
[14] A. Reyhani-Masoleh and M.A. Hasan, "Efficient Digit-serial Normal Basis Multipliers over Binary Extension Fields," *ACM Trans. Embedded Computing Systems,* vol. 3, no. 3, pp. 575-592, Aug. 2004.
[15] 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," *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES),* pp. 76-91, Aug. 2004.
[16] A. Reyhani-Masoleh, "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," *IEEE Trans. Computers,* vol. 55, no. 1, pp. 34-47, Jan. 2006.
[17] A.H. Namin, H. Wu, and M. Ahmadi, "A Word-Level Finite Field Multiplier Using Normal Basis," *IEEE Trans. Computers,* vol. 60, no. 6, pp. 890-895, June 2010.
[18] C. Lee and P. Chang, "Digit-Serial Gaussian Normal Basis Multiplier over $GF(2^m)$ Using Toeplitz Matrix-Approach," *Proc. Int'l Conf. Computational Intelligence and Software Eng. (CiSE),* pp. 1-4, 2009.
[19] Ç. K. Koç and B. Sunar, "An Efficient Optimal Normal Basis Type II Multiplier over $GF(2^m)$," *IEEE Trans. Computers,* vol. 50, no. 1, pp. 83-87, Jan. 2001.
[20] M. Hasan, M. Wang, and V. Bhargava, "A modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Trans. Computers,* vol. 42, no. 10, pp. 1278-1280, Oct. 1993.
[21] A. Reyhani-Masoleh and M.A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$," *IEEE Trans. Computers,* vol. 51, no. 5, pp. 511-520, May 2002.
[22] L. Gao and G.E. Sobelman, "Improved VLSI Designs for Multiplication and Inversion in $GF(2^m)$ over Normal Bases," *Proc. IEEE 13th Ann. Int'l ASIC/SOC Conf.,* pp. 97-101, 2000.
[23] K. Järvinen and J. Skyttä, "On Parallelization of High-Speed Processors for Elliptic Curve Cryptography," *IEEE Trans. Very Large Scale Integration (VLSI) Systems,* vol. 16, no. 9, pp. 1162-1175, Sept. 2008.
[24] C.H. Kim, S. Kwon, and C.P. Hong, "FPGA Implementation of High Performance Elliptic Curve Cryptographic Processor over $GF(2^{163})$," *J. System Architecture,* vol. 54, no. 10, pp. 893-900, 2008.
[25] 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),* pp. 25-40, June 2010.
[26] C.-P. Schnorr, "Efficient Signature Generation by Smart Cards," *J. Cryptology,* vol. 4, no. 3, pp. 161-174, 1991.
[27] A. Menezes, I. Blake, S. Gao, R. Mullin, S. Vanstone, and T. yaghoobian, *Applications of Finite Fields.* Kluwer Academic Publisher, 1993.
[28] C.-Y. Lee, "Concurrent Error Detection Architectures for Gaussian Normal Basis Multiplication over $GF(2^m)$," *Integration, the VLSI J.,* vol. 43, no. 1, pp. 113-123, 2010.
[29] M. Elia and M. Leone, "On the Inherent Space Complexity of Fast Parallel Multipliers for $GF(2^m)$," *IEEE Trans. Computers,* vol. 51, no. 3, pp. 346-351, Mar. 2002.
[30] C. Wang and D. Pei, "A VLSI Design for Computing Exponentiations in $GF(2^m)$ and Its Application to Generate Pseudorandom Number Sequences," *IEEE Trans. Computers,* vol. 39, no. 2, pp. 258-262, Feb. 1990.
[31] C. Lee, J. Lin, and C. Chiou, "Scalable and Systolic Architecture for Computing Double Exponentiation over $GF(2^m)$," *Acta Applicandae Mathematicae,* vol. 93, no. 1, pp. 161-178, 2006.
[32] J.H. Cheon, S. Jarecki, T. Kwon, and M.-K. Lee, "Fast Exponentiation Using Split Exponents," *IEEE Trans. Information Theory,* vol. 57, no. 3, pp. 1816-1826, Mar. 2011.
[33] J. Fan, D. Bailey, L. Batina, T. Guneysu, C. Paar, and I. Verbauwhede, "Breaking Elliptic Curves Cryptosystems using Reconfigurable Hardware," *Proc. 20th Int'l Conf. Field Programmable Logic and Applications (FPL),* pp. 133-138, 2010.
[34] Certicom, "Certicom ECC Chalenge," www.certicom.com, 1997.
[35] T. Itoh and S. Tsujii, "A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases," *Information Computing,* vol. 78, no. 3, pp. 171-177, 1988.

Reza Azarderakhsh received the BSc degree in electrical and electronic engineering in 2002, the MSc degree in computer engineering from Sharif University of Technology, Iran, in 2005, and the PhD degree in electrical and computer engineering from the University of Western Ontario in 2011. In 2006, he was with Polytechnic University of Turin, Italy for a special double degree program in electrical engineering with Sharif University of Technology. In 2011, he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Industrial Research and Development Fellowship. In September 2011, he joined the Department of Electrical and Computer Engineering of the University of Western Ontario, as a limited duties instructor. Currently, he is a postdoctoral fellow in the Center for Applied Cryptographic Research (CACR) and Department of Combinatorics and Optimization at the University of Waterloo. His current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography. He is a student member of the IEEE.

Arash Reyhani-Masoleh received the BSc degree in electrical and electronic engineering from Iran University of Science and Technology in 1989, the MSc degree in electrical and electronic engineering from the University of Tehran in 1991, both with the first rank, and the PhD degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he was with the Centre for Applied Cryptographic Research, University of Waterloo, where he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship in 2002. In October 2004, he joined the Department of Electrical and Computer Engineering, University of Western Ontario, London, Canada, where he is currently a tenured associate professor. His current research interests include algorithms and VLSI architectures for computations in finite fields, fault-tolerant computing, and error-control coding. He has been awarded a NSERC Discovery Accelerator Supplement (DAS) in 2010. Currently, he serves as an associate editor for Integration, the VLSI Journal (Elsevier). He is a member of the IEEE and the IEEE Computer Society.