A Modified Low Complexity Digit-Level Gaussian Normal Basis Multiplier

Reza Azarderakhsh and Arash Reyhani-Masoleh

Department of Electrical and Computer Engineering
The University of Western Ontario
London, ON, CANADA, N6A 5B9
{razarder and areyhani}@uwo.ca

Abstract. Gaussian normal bases have been included in a number of standards, such as IEEE [1] and NIST [2] for elliptic curve digital signature algorithm (ECDSA). Among different finite field operations used in this algorithm, multiplication is the main operation. In this paper, we consider type T Gaussian normal basis (GNB) multipliers over $GF(2^m)$, where m is odd. Such fields include five binary fields recommended by NIST for ECDSA. A modified digit-level GNB multiplier over $GF(2^m)$ is proposed in this paper. For $T > 2$, a complexity reduction algorithm is proposed to reduce the number of XOR gates without increasing the gate delay of the digit-level multiplier. The original and modified digit-level GNB multipliers are implemented on the Xilinx® Virtex5™ FPGA family for different digit sizes. It is shown that the modified digit-level GNB multiplier requires lower space complexity with almost the same delay as compared to the original type T, $T > 2$, GNB multiplier. Moreover, the bit-parallel GNB multiplier obtained from the proposed modified digit-level multiplier has the least space and time complexities among the existing fast bit-parallel type T GNB multipliers for $T > 2$.

Keywords: Finite field, Gaussian normal basis, digit-serial multiplier, complexity reduction.

1 Introduction

Elliptic curve cryptosystem, which is proposed independently by Miller [3] and Koblitz [4], requires extensive finite field operations for the point multiplication. Multiplication is the main operation and its structure depends strongly on the representation of the field element. There are a number of ways to represent field elements. Among them, the most common bases are the polynomial basis and the normal basis representations [1]. In normal basis representation, squaring of a field element is free in hardware. Recently, implementation of the point multiplication for elliptic curve cryptography (ECC) using normal basis has received attention in the literature, see for example [5], [6] and [7].

The first normal basis multiplier over $GF(2^m)$ was invented by Massey and Omura [8]. This bit-serial multiplier, which has a parallel-in serial-out structure,

generates a bit of the result in each clock cycle. Therefore, the coordinates of the multiplication are generated after m clock cycles with the least complexity. There are also other bit-serial multipliers with parallel outputs, see for example [9], [10]. To make a fast hardware implementation, a bit-parallel multiplier is proposed in [11] by having m copies of identical bit-serial structure of [8] with shifted inputs. In such a multiplier, once $2m$ bits of two inputs are received, m bits of the product are obtained after propagation delay through gates. Various efficient bit-parallel architecture for normal basis multiplication over $GF(2^m)$ have been developed in the literature, see for example [11], [12] and [13] for arbitrary normal basis as well as [14], [15], [16], and [17] for special classes of normal basis.

Bit-parallel multipliers require a lot of silicon area and it is impractical for resource constrained environments such as smart cards. To obtain an optimum multiplier for such applications, a digit-level multiplier can be utilized, where the digit size can be chosen depending on the available resources. Using a digit-level multiplier allows the designers to trade-off between speed and area. Among different digit-level normal basis multipliers available in the literature, the ones with the parallel outputs run at much higher frequency than the other ones. Such a multiplier is proposed in [15] and [7] for GNB over $GF(2^m)$, where m is odd.

A special classes of normal basis called GNBs, have been included in the recent standards, such as IEEE and NIST for ECDSA. In this paper, a complexity reduction algorithm is proposed to reduce the number of XOR gates for the original parallel-output type T digit-level GNB multiplier proposed in [15] and [7] for $T > 2$. This algorithm uses sub-expression sharing without increasing the gate delay of the multiplier. It is noted that no such common terms is obtained for $T = 2$. Thus, the algorithm is coded using MATLAB for the $GF(2^{163})$ ($T = 4$) and $GF(2^{283})$ ($T = 6$) finite fields in terms of different digit sizes. Then, based on the results obtained from the algorithm, the original digit-level multiplier structure is modified. The modified GNB multiplier requires fewer number of XOR gates without impacting the gate delay. Both the original $GF(2^{163})$ and $GF(2^{283})$ multipliers and the modified ones are compared in terms of number of XORs for different digit sizes. To obtain the actual implementation results, the original and modified structures are coded in VHDL and they are implemented on a Xilinx® Virtex5™ field-programmable gate array (FPGA) device for different digit sizes. The comparison results show that the modified structure outperforms the original one in terms of area without significantly affecting the multiplication time. It is also shown that for the highest digit size, the bit-parallel multiplier obtained from the modified digit-level multiplier requires the least number of XOR gates with the same gate delay compared to the existing fast GNB multipliers.

The organization of the remaining parts of this paper is as follows. In Section 2, we state the preliminaries required in this paper. Also, the original digit-level Gaussian normal basis multiplier with parallel output is presented in this section. A modified version of this multiplier is proposed in Section 3 using a complexity

reduction algorithm. Moreover, in this section a bit-parallel GNB multiplier obtained from the proposed digit-level GNB multiplier is presented and compared with its counterparts in terms of time and area complexities. Results of the hardware implementations of the proposed multiplier on the Xilinx® Virtex5™ FPGA are presented in Section 4. We finally conclude the paper in Section 5.

2 Preliminaries

It is well known that there is always a normal basis $N = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$, for a finite field $GF(2^m)$ over $GF(2)$ for any positive integer m, where β is called normal element [18]. The elements of N are linearly independent and each element, say $A = (a_0,\ a_1, \cdots, a_{m-1})$, can be represented as a linear combination of the elements in N, as $A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$, where coefficients $a_i \in GF(2)$, $0 \leq i \leq m-1$, denote the coordinates of A. The merit of the normal basis is that, squaring of an element A in the normal basis representation is a right cyclic shift of its coordinates, i.e., $A^2 = (a_{m-1},\ a_0,\ a_1, \cdots, a_{m-2})$, and it is free in hardware.

Definition 1. *Let $p = mT + 1$ be a prime number and $gcd(mT/k,\ m) = 1$, where k is the multiplication order of 2 module p. Then, the normal basis $N = \{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$ over $GF(2^m)$ is called the Gaussian normal basis of type T, $T > 0$.*

It is noted that the GNBs exist over $GF(2^m)$ whenever m is not divisible by 8 [1]. In this paper, we only consider the GNBs with odd values of m. This implies that T is an even number. It is noted that such GNBs are important since they include the five binary fields, i.e., $m \in \{163, 233, 283, 409, 571\}$, recommended by NIST for ECDSA [2]. The corresponding types for these fields are $T = 4, 2, 6, 4$, and 10, respectively.

2.1 Normal Basis Multiplication

Let $A = (a_0,\ a_1, \cdots, a_{m-1}) = \sum_{i=0}^{m-1} a_i \beta^{2^i}$ and $B = (b_0,\ b_1, \cdots, b_{m-1}) = \sum_{j=0}^{m-1} b_j \beta^{2^j}$ be two field elements over $GF(2^m)$. Let $C \in GF(2^m)$ be their product, i.e., $C = (c_0,\ c_1,\ \cdots,\ c_{m-1}) = AB = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i+2^j}$. Let us represent the field element $\beta^{2^i+2^j} \in GF(2^m)$, $0 \leq i, j \leq m-1$, with respect to N as $\beta^{2^i+2^j} = \sum_{l=0}^{m-1} \mu_{i,j}^{(l)} \beta^{2^l}$.Then, one can find C as

$$C = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \sum_{l=0}^{m-1} \mu_{i,j}^{(l)} \beta^{2^l} = \sum_{l=0}^{m-1} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \mu_{i,j}^{(l)} \beta^{2^l}. \quad (1)$$

By representing C with respect to N, i.e., $C = \sum_{l=0}^{m-1} c_l \beta^{2^l}$, and equating it with (1), the l-th coordinate of C can be written as $c_l = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \mu_{i,j}^{(l)}$. Then, it can be written in a matrix form as

$$c_l = \underline{a} \mathbf{M}^{(l)} \underline{b}^{tr},\ 0 \leq l \leq m-1, \quad (2)$$

where $\mathbf{M}^{(l)} = [\mu_{i,j}^{(l)}]_{i,j=0}^{m-1}$, $\mu_{i,j}^{(l)} \in GF(2)$, $0 \leq i, j \leq m-1$, $\underline{a} = [a_0, a_1, \cdots, a_{m-1}]$ and $\underline{b}^{tr}$ denotes the matrix transpose of row vector $\underline{b} = [b_0, b_1, \cdots, b_{m-1}]$. In (2), $\mathbf{M}^{(l)}$ is obtained from the l-fold right and down circular shifts of the *multiplication matrix* $\mathbf{M} = \mathbf{M}^{(0)}$. The computation of entries of $\mathbf{M}$ can be found from [1]. Massey and Omura in [8] proposed the bit-serial multiplier by implementing (2) for one coordinate, say $c_0 = \underline{a}\mathbf{M}\underline{b}^{tr} = F(A, B)$. Then, the lth coordinate of C can be obtained by cyclic shifts of the coordinates of A and B, i.e., $c_l = F(A \ll l,\ B \ll l)$ [8].

The number of non-zero entries in the multiplication matrix $\mathbf{M} = \mathbf{M^{(0)}}$ in (2) is called the complexity of the normal basis and is denoted by C_N [19]. This can be used to estimate the area complexity of hardware implementation of the multiplier. Gao et al. in [20] proved that $C_N \geq 2m-1$. The normal basis is said to be optimal if $C_N = 2m-1$. The optimal normal bases are extended to another class of low complexity normal basis called Gaussian normal basis by Ash et. al [21]. For type T GNB, $T \geq 2$, the complexity of multiplication matrix $\mathbf{M}$ satisfies $C_N \leq mT - 1$ [21]. A slightly tighter upper bound for C_N is found in [10] as $C_N = mT - T + 1$. Therefore, if there is no optimal normal basis for a given m, the GNB with the least value of T is an alternative for choosing normal bases.

2.2 Digit-Level Gaussian Normal Basis Multiplier with Parallel Output

Let $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ be the GNB elements over $GF(2^m)$, and let d, $1 \leq d \leq m$, be the digit size. Reyhani-Masoleh in [15] and Kim et al. in [7] proposed a digit-level Gaussian normal basis multiplier with parallel output (DLGMp). It requires q, $1 \leq q \leq m$, clock cycles to generate all m coordinates of $C = AB$ simultaneously at the end of the final clock cycle. The original multiplier structure of DLGMp is shown in Figure 1. Let $X = (x_0, x_1, \cdots, x_{m-1})$ and $Y = (y_0, y_1, \cdots, y_{m-1})$ be the input registers of this multiplier. Then, it implements [15]

$$J(X,\ Y) = \sum_{k=0}^{m-1} x_{m-k} s_0^{'}(k,\ Y)\beta^{2^i}, \tag{3}$$

where

$$s_0^{'}(k,\ Y) = \sum_{i \in R_k} y_{i-k}, \tag{4}$$

and R_k is a set containing the locations of non-zero entries of row $2k$, $0 \leq 2k \leq m-1$, of the multiplication matrix $\mathbf{M} = \mathbf{M^{(0)}}$ defined in (2). Based on the properties of $\mathbf{M}$ for GNB, one can find $s_0^{'}(0,\ Y) = y_1$ and $s_0^{'}(k,\ Y) = s_0^{'}(m-k,\ Y)$, $1 \leq k \leq \frac{m-1}{2}$ [15]. Also, it is shown in [10] and [15] that the number of elements in R_k is even and less than or equal to T, i.e., $|R_k| \leq T$. The J block in Figure 1 performs (3) using m AND gates. For the multiplication operation,

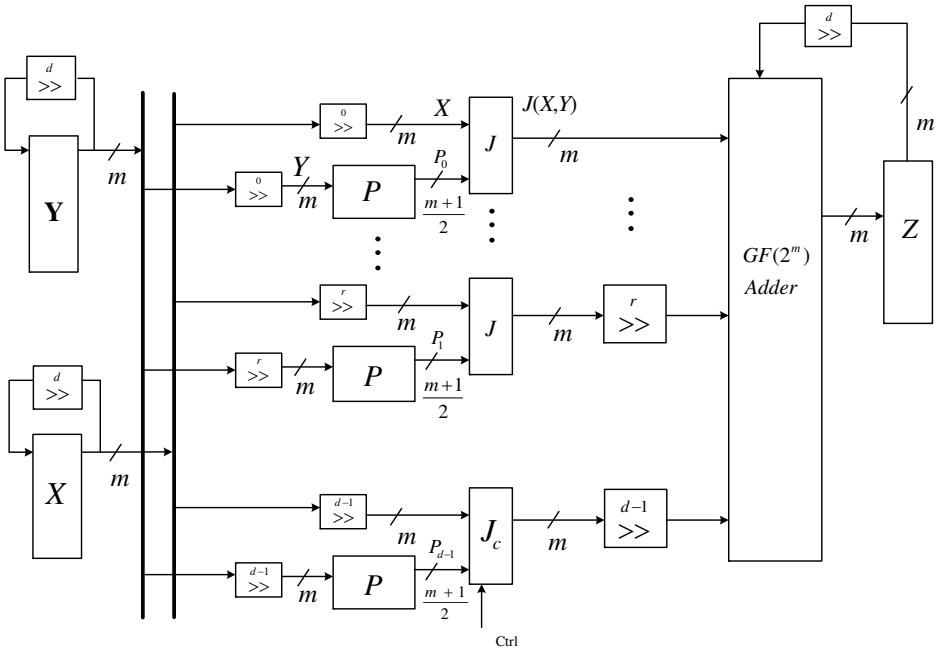


Fig. 1: Digit-serial Gaussian normal basis multiplier proposed in [15], [7], where the i-fold right cyclic shift is denoted by $\overset{i}{\gg}$ and r is a number $0 \leq r \leq d-1$ such that $m = qd - r$.

the registers X and Y of this figure are initially loaded by the coordinates of A and B, respectively. Also, the output register Z should be cleared before starting the multiplication operation. Then, after q clock cycles, the output register Z contains the coordinates of $C = AB$. In the following section, we modify this multiplier to reduce the number of XOR gates.

3 Modified Digit-Level GNB Multiplier

The number of XOR gates of the DLGMp multiplier presented in the previous section can be reduced by reusing the common terms appeared at the outputs of the P blocks. The complexity reduction scheme presented in [15] cannot be applied for a practical field, such as, $GF(2^{163})$ and $GF(2^{283})$. For the $GF(2^7)$ example used in [15], the P block is optimized first and then the same block is copied for all P blocks used in the multiplier. It is interesting to note that for type 4 GNB over $GF(2^{163})$, no common pair can be found in the P block of Figure 1 if one applies the method presented in [15]. For this purpose, we modify this multiplier by replacing all P blocks that generate $P_1, \cdots, P_d$ in Figure 1 with one block. As seen in Figure 1, the number of outputs of an unoptimized P block in this figure is $\frac{m+1}{2}$. These are based on the following signals [15]

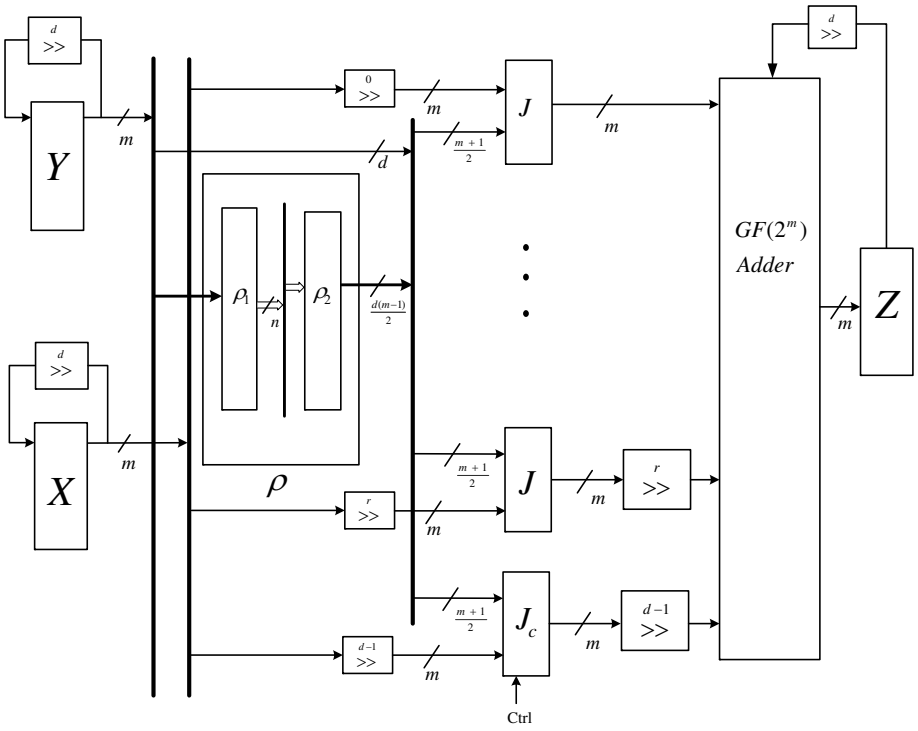


Fig. 2: Modified Digit-level Gaussian Normal Basis Multiplier (MDLGMp).

$$P_k(Y) = (y_{1-k},\ s_0^{'}(1, Y \ll k),\ s_0^{'}(2, Y \ll k), \cdots \\ , \cdots , s_0^{'}(\tfrac{m-1}{2},\ Y \ll k)),\ 0 \leq k \leq d-1, \tag{5}$$

for the P block that generates $P_k(Y)$. The modified digit-level multiplier is shown in Figure 2. The combination of all P blocks in Figure 1 is shown by ρ in Figure 2. All signals in (5) are used to build the block ρ in Figure 2. As shown in this figure, y_{1-k}s are removed from the block ρ. To reduce the complexity of the ρ block in Figure 2, we divide the ρ block in two blocks ρ_1 and ρ_2, where ρ_1 includes all common pairs used to generate all signals in (5). In the following section, a complexity reduction algorithm is presented for a given GNB to obtain the optimized blocks of ρ_1 and ρ_2 so that the time delay (in terms of gate delays) of the original block ρ is the same as the one in the modified multiplier, i.e., the addition of gate delays of the two blocks ρ_1 and ρ_2.

3.1 A Complexity Reduction Algorithm

In this section, an approach for reducing the area complexity of the modified digit-level GNB multiplier is proposed. It is noted that unlike the complexity

reduction schemes available in the literature, see for example [22], the proposed algorithm does not increase the gate delay of the modified structure as compared to the original one. The complexity reduction algorithm to reduce the number of XOR gates in the block ρ of Figure 2 is summarized as follows.

Input: The multiplication matrix $\mathbf{M}$ and digit size d for type T GNB over $GF(2^m)$.

Output: A pairset which contains all the pairs that should be implemented in the block ρ_1. This set will be used to obtain the formulations for the implementation of the modified multiplier.

1. Corresponding to the output signals of the P block in Figure 1, an $\frac{m-1}{2} \times T$ matrix denoted by $\mu = [\mu_k]_{k=1}^{\frac{m-1}{2}}$ is constructed, where μ_k is the row k, $1 \leq k \leq \frac{m-1}{2}$ of the matrix μ. The entries of μ_k are at most T integers in the range of $[0, m-1]$ and can be found from (4) which can be written as $s_0^{'}(k,\ Y) = \sum_{j \in \mu_k} y_j$, $1 \leq k \leq \frac{m-1}{2}$.
2. Based on the matrix μ and the given digit-size d, a matrix denoted by ρ is obtained by appending the $d-1$ matrices of $\mu - [i] \mod m$ to μ as follows:

$$\rho = \begin{bmatrix} \mu \\ \mu - [1] \bmod m \\ \mu - [2] \bmod m \\ \vdots \quad \vdots \\ \mu - [d-1] \bmod m \end{bmatrix}_{(d \times \frac{m-1}{2}) \times T,} \tag{6}$$

   where $[i]$, $1 \leq i \leq d$, denotes an $\frac{m-1}{2} \times T$ matrix whose all entries are i.
3. Let ρ_i be a set which contains the entries in row i of the matrix ρ. Then, all signals

$$s_j = \sum_{j \in \rho_i} y_j,\ 1 \leq j \leq d\frac{(m-1)}{2} \tag{7}$$

   should be implemented by the block ρ shown in Figure 2.
4. We want to find the common addition pairs to realize (7) with the least number of XOR gates without changing the delay of the modified multiplier as compared with the original one. Therefore, a pairset is generated to form all pairs that should be implemented in the block ρ_1. This set initially contains all pairs with only two entries in the rows of the matrix ρ. We update the ρ matrix by removing such pairs from the matrix. Then, go to Step 3.
5. The scheme will be terminated if no common terms will be obtained.
6. Finally, based on common pairs stored in the pairset, the ρ_1 inside the ρ is generated. By reusing the output of the block ρ_1, we can generate all signals from the block ρ_2 in Figure 2.

In the following section, we present an illustrative example for the proposed complexity reduction algorithm.

3.2 An Example over $GF(2^7)$

To better understand the complexity reduction algorithm, we illustrate an example for the proposed algorithm for type 4 digit-level multiplier over $GF(2^7)$ when the digit-size is $d = m = 7$. The matrix $\mathbf{M}$ for type 4 GNB over $GF(2^7)$ is

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

The matrix μ can be generated according to the output of the P blocks in Figure 1 as $s_0^{'}(1,Y) = y_{1-1} + y_{3-1} + y_{4-1} + y_{5-1} = y_0 + y_2 + y_3 + y_4$, $s_0^{'}(2,Y) = y_{2-2} + y_{6-2} = y_0 + y_4$, and $s_0^{'}(3,Y) = y_{1-3} + y_{4-3} + y_{5-3} + y_{6-3} = y_5 + y_1 + y_2 + y_3$. Then μ can be written as

$$\mu = \begin{pmatrix} 0\,2\,3\,4 \\ 0\,4\,-\,- \\ 5\,1\,2\,3 \end{pmatrix}_{3\times 4}.$$

$$\rho = \begin{pmatrix} 0\,2\,3\,4 \\ 0\,4\,-\,- \\ 5\,1\,2\,3 \\ 6\,1\,2\,3 \\ 6\,3\,-\,- \\ 4\,0\,1\,2 \\ 5\,0\,1\,2 \\ 5\,2\,-\,- \\ 3\,6\,0\,1 \\ 4\,6\,0\,1 \\ 4\,1\,-\,- \\ 2\,5\,6\,0 \\ 3\,5\,6\,0 \\ 3\,0\,-\,- \\ 1\,4\,5\,6 \\ 2\,4\,5\,6 \\ 2\,6\,-\,- \\ 0\,3\,4\,5 \\ 1\,3\,4\,5 \\ 1\,5\,-\,- \\ 6\,2\,3\,4 \end{pmatrix}_{21\times 4} \quad \text{Pairset1} = \begin{Bmatrix} y_{04} \\ y_{63} \\ y_{52} \\ y_{41} \\ y_{30} \\ y_{26} \\ y_{15} \end{Bmatrix} \quad \rho^{(1)} = \begin{pmatrix} 0\,2\,3\,4 \\ 5\,1\,2\,3 \\ 6\,1\,2\,3 \\ 4\,0\,1\,2 \\ 5\,0\,1\,2 \\ 3\,6\,0\,1 \\ 4\,6\,0\,1 \\ 2\,5\,6\,0 \\ 3\,5\,6\,0 \\ 1\,4\,5\,6 \\ 2\,4\,5\,6 \\ 0\,3\,4\,5 \\ 1\,3\,4\,5 \\ 6\,2\,3\,4 \end{pmatrix} \rho^{(2)} = \begin{pmatrix} 2\,3 \\ 1\;\;3 \\ \underline{1}\,\underline{2} \\ \;\;\underline{1}\,\underline{2} \\ \underline{0}\,\underline{1} \\ \;\;\underline{0}\,\underline{1} \\ \underline{6}\,\underline{0} \\ \;\;\underline{6}\,\underline{0} \\ 5\;\;0 \\ \;\;5\,6 \\ 4\;\;6 \\ \;\;4\,5 \\ 3\;\;5 \\ 2\;\;4 \end{pmatrix} \quad \text{Pairset2} = \begin{Bmatrix} y_{23} \\ y_{13} \\ y_{12} \\ y_{01} \\ y_{60} \\ y_{50} \\ y_{56} \\ y_{46} \\ y_{45} \\ y_{35} \\ y_{24} \end{Bmatrix}$$

Based on the digit-size $d = 7$ and the matrix $\mu_{(3\times 4)}$, the matrix $\rho_{(21\times 4)}$ can be generated corresponding the complexity reduction algorithm. One can obtain from the matrix $\rho_{(21\times 4)}$ in which 7 rows of the matrix have just two entries. Therefore, the pairs corresponding to these rows should be implemented as collected in the pairset1. The matrix ρ is updated to $\rho^{(1)}$ by deleting all the two entries mentioned in the pairset1. Then the elements of the pairset1 should be searched in $\rho^{(1)}$ and all common pairs are removed and $\rho^{(1)}$ is updated to $\rho^{(2)}$. This iteration is repeated until there is no rows with more than two entries. As a result, all the remaining pairs as mentioned in the pairset2 should

be implemented and repeated pairs (which are underlined in the updated $\rho^{(2)}$ matrix) are removed. The union of pairset1 and pairset2 includes the total of 18 pairs that should be implemented for the block ρ_1 as follows:

$$\begin{aligned}\text{pairset}=\{&y_{04}, y_{63}, y_{52}, y_{41}, y_{30}, y_{26}, y_{15}, y_{23}, y_{13}, y_{12}, y_{01}, y_{60},\\ &y_{50}, y_{56}, y_{46}, y_{45}, y_{35}, y_{24}\},\end{aligned}$$

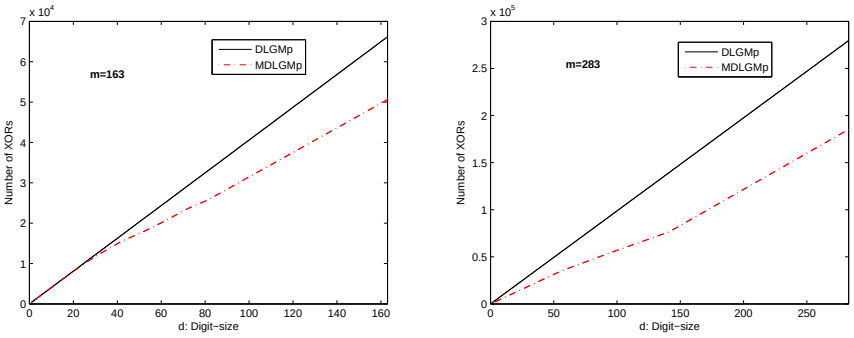
where $y_{ij} = y_i + y_j$. In addition to the implementation of the ρ block which requires 18 XOR gates, one need $d\frac{m-1}{2} - d = 14$ (as, $d = m$) extra XOR gates for the block ρ_2 to construct its outputs. Therefore, the total number of XOR gates required to implement the ρ block will be $18+14 = 32$, whereas the unoptimized P blocks need 49 XOR gates and the scheme proposed in [15] requires 35 XOR gates.

It is noted that the other complexity reduction algorithms available in the literature may result in fewer number of gates at the expense of more delay as the one proposed in this paper. To compare our complexity reduction algorithm with the one proposed in [22], we have applied the complexity reduction algorithm proposed in [22] for the block ρ of this example. It decreases the number of XORs to 23 with the increase of critical path delay to $8T_X$ (eight level of XOR gates). Note that our scheme for this block results in the complexity of 32 XOR gates with the same critical path delay as the original one, i.e., $2T_X$.

3.3 Simulation Results for the Digit-Level GNB Multipliers over $GF(2^{163})$ and $GF(2^{283})$

To evaluate the efficiency of our complexity reduction algorithm, a MATLAB code is written to generate common pairs and signals used in the blocks ρ_1 and ρ_2 of Figure 2. It is noted that for type 2 GNB which is a type 2 optimal normal basis over $GF(2^m)$, there is no common terms to be reused in the block ρ. Therefore, the algorithm presented in this paper cannot reduce the number of XOR gates for $T = 2$. The simulation results of the algorithm for the modified digit-level GNB multipliers (MDLGMp) over $GF(2^{163})$ and $GF(2^{283})$ are obtained and plotted in Figures 3a and 3b. In these figures, we plot the number of required XOR gates versus the digit size for the fields $GF(2^{163})$ ($T = 4$) and $GF(2^{283})$ ($T = 6$) recommended by NIST for ECDSA [2] as compared to ones of the original digit-level multiplier with parallel-output (DLGMp). For a given number of clock cycle, q, $1 \leq q \leq m$, the least value of digit sizes in the form of $d = \left\lceil \frac{m}{q} \right\rceil$, $1 \leq d \leq m$, is implemented so that the area complexity is optimized for both multipliers.

From Figures 3a and 3b, one can see that as the digit size increases, more common pairs will be found. As an example, in Figure 3a for the digit size $d = m = 163$, the total number of XOR gates required in the original DLGMp is 66178 gates whereas, the modified one, requires 50400 XOR gates for $GF(2^{163})$. It means that the complexity of the proposed MDLGMp is about



(a) (b)

Fig. 3: Comparison between the number of XOR gates required in the DLGMp and the MDLGMp, for (a): $m = 163$ ($T = 4$), (b): $m = 283$ ($T = 6$).

24% less than the original multiplier. More reduction can be found in Figure 3b for the $GF(2^{283})$ with $d = m = 283$. The number of XOR gates needed by the original DLGMp is 279,604, whereas the proposed MDLGMp requires 185,375 XOR gates which is about 34% less than that of the original multiplier.

The formulations for the output signals of the blocks ρ_1 and ρ_2 are coded in VHDL to obtain the actual FPGA implementation results. The implementation results are presented in Section 4.

3.4 An Extension to Bit-Parallel GNB Multiplier

To obtain the bit-parallel multiplier, one can implement (2) in hardware for all c_l, $0 \leq l \leq m-1$. Thus, the hardware architecture of a bit-parallel multiplier is obtained by implementing m copies of identical structures used for c_0 with cyclic shifts of their inputs [11]. In this section, we extend the modified digit-level multiplier for $d = m$ to obtain a new bit-parallel GNB multiplier over $GF(2^m)$. Then, its complexities are obtained and compared with the ones of its counterparts.

Let n denote the total number of common pairs. Thus the block ρ_1 contains at most n XOR gates with the delay of an XOR gate. In the worst case, all combinations of two coordinates of A, i.e., $\binom{m}{2} = \frac{m(m-1)}{2}$ combinations, are required in the block ρ_1 for the bit-parallel multiplier and so, $n \leq \frac{m(m-1)}{2}$.

The block ρ_2 consists of XOR gates for the GNB, with $T > 2$. This is because there is no row in $\mathbf{M}$ with the number of 1s greater than 2 for type 2 GNB. Thus, for $T = 2$, $n = \frac{m(m-1)}{2}$ and the block ρ_2 connects its input bus to the next bus without using any XOR gates.

The exact complexities of ρ_1 and ρ_2 depend on the GNB. However, one can find the upper bound for the number of XOR gates and time delay of this structure as follows.

Proposition 1. *For Type T GNB over $GF(2^m)$, the proposed bit-parallel Gaussian normal basis multiplier architecture requires m^2 AND gates and at most $(T+4)(\frac{m(m-1)}{4})$ XOR gates with the critical path delay of*

$$T_C = T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X, \tag{8}$$

where T_A and T_X are the time delay of a two-input AND gate and an XOR gate, respectively.

Proof. Let n be the the number of XOR gates which is the number of the pairs required to construct the block ρ_1. As mentioned earlier, one can see that the upper bound of n can be found from $n \leq \binom{m}{2} = \frac{m(m-1)}{2}$. Thus, the block ρ_1 contains at most $\frac{m(m-1)}{2}$ XOR gates. It is noted that each output of the block ρ_2 is modulo 2 addition of at most T coordinates of A which can be obtained by adding at most $\frac{T}{2}$ signals from the output of ρ_1. Therefore, the number of XOR gates required to construct the block ρ_2 of the bit-parallel multiplier is $(\frac{T}{2}-1)(\frac{m-1}{2}) \times m = \frac{m(m-1)(T-2)}{4}$. The rest of Figure 2 requires m^2 AND gates and $m(m-1)$ XOR gates to implement all J blocks and the $GF(2^m)$ adder. By adding the number of XOR gates in the ρ_1, ρ_2 and other blocks, one can obtain the upper bound for the total number of XOR gates as $\frac{m(m-1)}{2} + \frac{m(m-1)(T-2)}{4} + m(m-1) = (T+4)(\frac{m(m-1)}{4})$.

The critical-path delay of the proposed architecture can be obtained by adding the delays of the three blocks of ρ_1, ρ_2, J, and the $GF(2^m)$ adder which are T_X, $\lceil \log_2 \frac{T}{2} \rceil T_X$, T_A, and $\lceil \log_2 m \rceil T_X$, respectively. This results in the total delay of $T_X + \lceil \log_2 \frac{T}{2} \rceil T_X + T_A + \lceil \log_2 m \rceil T_X = T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$, which completes the proof.

3.5 Comparison

The time and area complexities of the proposed bit-parallel GNB multiplier and the previous schemes are compared in Table 1 for general and special values of T. As shown in the table, the critical path delay of the proposed multiplier matches the fastest results available in the literature. For type $T = 2$ GNB, the number of XOR gates also matches the fastest result available in the open literature, i.e., $1.5m(m-1)$. However, it is much greater than the sub-quadratic results proposed in [17] and [16] which require much higher delay as compared to the one proposed here. It is interesting to note that for $T > 2$, the proposed multiplier outperforms its counterparts with the same delay in terms of number of XOR gates as shown in this table. It is noted the number of XOR gates required for the new bit-parallel GNB multiplier is still greater than the one required for the polynomial basis.

It should be noted that, to obtain the exact number of XOR gates for a given GNB, the exact value of n should be obtained by simulations. Using the complexity reduction algorithm proposed in Section 3.1, a comparison between the number of XOR gates of bit-parallel GNB multipliers is illustrated in Table 2 for $GF(2^{163})$ and $GF(2^{283})$ fields recommended by NIST for ECDSA.

Table 1: Area and time complexity comparison of bit-parallel GNB multipliers over $GF(2^m)$. Note that for Type T GNB: $C_N \leq Tm - T + 1$.

Multiplier	$T \geq 2$		
	#AND	#XOR	Critical path
Massey & Omura [8]	m^2	$m(C_N - 1)$	$T_A + \lceil \log_2 C_N \rceil T_X$
Gao & Sobelman[12]	m^2	$m(C_N - 1)$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$
Reyhani-Masoleh & Hasan [13]	m^2	$\leq \frac{m}{2}(C_N + m - 2)$	$T_A + (\lceil \log_2 (C_N + 1) \rceil) T_X$
DLGMp [15], [7] ($d = m$)	m^2	$\leq \frac{m}{2}(C_N + m)$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 (m) \rceil) T_X$
DLGMs [15] ($d = m$)	m^2	$\leq \frac{m(m-1)}{2}(T + 1)$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 (m) \rceil) T_X$
This work	m^2	$\leq (\frac{m(m-1)}{4})(T + 4)$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 (m) \rceil) T_X$
	T=2		
[8,12]	m^2	$2m(m - 1)$	$T_A + \lceil \log_2 (2m - 1) \rceil T_X$
Koc & Sunar [14]	m^2	$1.5m(m - 1)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
Fan & Hasan [16]	$2m^{1.6}$	$11m^{1.6} - 12m + 1$	$T_A + (2 \log_2 m + 1) T_X$
Gathen et. al [17]	$2m^{1.6}$	$7.6m^{1.6} + \mathcal{O}(m \log m)$	$T_A + (2 \log_2 m + 1) T_X$
[13,15,7], This work	m^2	$1.5m(m - 1)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
	T=4		
[8], [12]	m^2	$4m^2 - 4m$	$T_A + (2 + \lceil \log_2 (m) \rceil) T_X$
Reyhani-Masoleh & Hasan [13]	m^2	$2.5m^2 - 4.5m$	$T_A + \lceil 1 + \log_2 (2m - 1) \rceil T_X$
DLGMp [15], [7] ($d = m$)	m^2	$2.5m^2 - 1.5m$	$T_A + (2 + \lceil \log_2 (m) \rceil) T_X$
DLGMs [15] ($d = m$)	m^2	$2.5m^2 - 2.5m$	$T_A + (2 + \lceil \log_2 (m) \rceil) T_X$
This work	m^2	$\leq 2m^2 - 2m$	$T_A + (2 + \lceil \log_2 (m) \rceil) T_X$
	T=6		
[8], [12]	m^2	$6m^2 - 6m$	$T_A + (3 + \lceil \log_2 (m) \rceil) T_X$
Reyhani-Masoleh & Hasan [13]	m^2	$3.5m^2 - 3.5m$	$T_A + (\lceil \log_2 (6m - 4) \rceil) T_X$
DLGMp [15], [7] ($d = m$)	m^2	$3.5m^2 - 2.5m$	$T_A + (3 + \lceil \log_2 (m) \rceil) T_X$
DLGMs [15] ($d = m$)	m^2	$3.5m^2 - 3.5m$	$T_A + (3 + \lceil \log_2 (m) \rceil) T_X$
This work	m^2	$\leq 2.5m^2 - 2.5m$	$T_A + (3 + \lceil \log_2 (m) \rceil) T_X$

Table 2: Comparison between bit-parallel GNB multipliers for $GF(2^{163})$ and $GF(2^{283})$.

m	T	n	Number of XORs in DLGM for $d = m$ [15]	Number of XOR gates used in this work
163	4	10791	66178	50400
283	6	25763	279604	185375

4 FPGA Implementations

The architectures described in Sections 2.2 and 3 are written in VHDL. We have implemented the original (DLGMp) and the modified digit-level multipliers (MDLGMp) on the Xilinx® Virtex5™ FPGA family with target device

Table 3: FPGA implementation results for propagation delay (in terms of nano second) and area (in terms of number of slices) for different digit sizes with $m = 163$ and $T = 4$. Target device is Xilinx xc5vlx330-2ff1760.

Digit size (d)	# of cycles (q)	Delay [ns]		Area [# of Slice LUTs]	
		DLGMp	MDLGMp	DLGMp	MDLGMp
1	163	2.8	2.8	1221	1221
2	82	3.1	3.1	1282	1280
3	55	3.1	3.1	1347	1346
4	41	3.4	3.4	1406	1406
5	33	3.5	3.6	1564	1565
6	28	4.1	4.2	1751	1750
7	24	3.8	3.8	1960	1960
8	21	3.7	3.8	2104	2104
9	19	4.3	4.4	2157	2157
10	17	4.2	4.2	2309	2309
11	15	4.2	4.2	2385	2385
12	14	4.5	4.5	2567	2567
13	13	4.6	4.6	2785	2780
14	12	4.5	4.6	2852	2850
15	11	4.8	4.7	2923	2923
17	10	4.9	4.9	3164	3164
19	9	4.9	4.9	4048	4045
21	8	5.4	5.5	4146	4140
24	7	5.6	5.7	4593	4385
28	6	5.7	5.7	4730	4652
33	5	5.8	5.8	5288	5023
41	4	6.1	6.1	6129	5633
55	3	6.4	6.5	8115	6091
82	2	7.3	7.5	11187	7321
163	1	11.5	11.9	22917	14238

xc5vlx330-2ff1760 for $GF(2^{163})$ and $GF(2^{283})$ fields. Correctness of the implementations is verified by performing functional simulations using the Quartus® II software. We have synthesized both multipliers for several different digit sizes d, $1 \leq d \leq m$ using Xilinx synthesis technology (XST), and the timing analysis results via Xilinx ISE-9.1.03i after place and route (PAR) are illustrated in Tables 3 and 4 for $GF(2^{163})$ and $GF(2^{283})$, respectively. As seen in the tables, large digit sizes require more area in terms of number of slice look up tables (LUTs). Therefore, we chose the digit size, d, in such a way to decrease the critical path delay while increasing the area. Note that other values for d only increases area without decreasing latency. Our presented multiplier requires less area than the original one for the different digit size d. The total multiplication time can be calculated as the product of the minimum clock period and the number of clock cycles q presented in both tables. Obviously, as shown in the Tables 3 and 4, the time complexities of these structures are almost the same. It means that, our

Table 4: FPGA implementation results for propagation delay (in terms of nano second) and area (in terms of number of slice LUTs) for different digit sizes with $m = 283$ and $T = 6$. The target device is xc5vlx330-2ff1760.

Digit size (d)	# of cycles (q)	Delay [ns]		Area [# of Slice LUTs]	
		DLGMp	MDLGMp	DLGMp	MDLGMp
1	283	3.4	3.4	2118	1985
2	142	3.7	3.8	2252	2088
3	95	3.9	3.9	2388	2156
4	71	4.1	4.1	2603	2437
5	57	4.4	4.5	2829	2603
6	48	4.5	4.6	3216	2714
7	41	4.4	4.5	3358	2937
15	19	4.7	4.8	5803	3986
16	18	5.1	5.2	6086	4105
19	15	5.7	5.7	6309	4387
22	13	5.8	5.8	6429	4427
24	12	8.7	8.9	7039	4938
26	11	8.6	8.5	7325	5183
29	10	8.3	8.4	7723	5563
35	9	8.3	8.5	9398	6769
71	4	11.4	11.4	17224	10345
142	2	12.4	12.3	31395	22512

proposed multiplier reduces the required area having the same multiplication time.

5 Conclusions

We have proposed a modified architecture for digit-level Gaussian normal basis multiplier over $GF(2^m)$. It is shown that this multiplier outperforms the original one in terms of number of XOR gates. Using a complexity reduction algorithm, the area complexity of the modified digit-level multiplier, is optimized. We have also presented a fast low complexity bit-parallel GNB multiplier over $GF(2^m)$. Its complexities have been derived and it is shown that it has fewer XOR gates for $T > 2$ and the same for $T = 2$ as compared to the fast GNB multipliers available in the literature. For practical applications, we have implemented the original and the modified digit-level GNB multipliers on the Xilinx® Virtex5™ FPGA family for different digit sizes. Our comparison results show that the modified digit-level GNB multiplier requires fewer area (slice LUTs) with almost the same delay as compared to the original one.

Acknowledgment

The authors of the paper would like to thank the reviewers for their comments. This work has been supported in part by an NSERC Discovery grant awarded to A. Reyhani-Masoleh.

References

1. IEEE Std 1363-2000: "IEEE Standard Specifications for Public-Key Cryptography", (January 2000)
2. U.S. Department of Commerce/NIST: Digital Signature Standards (DSS). Federal Information Processing Standards Publications, (2000)
3. Miller, V.S.: "Use of Elliptic Curves in Cryptography". In: LNCS 218 as Proceedings of Crypto '85, Springer Verlag 417–426 (1986)
4. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
5. Dimitrov, V.S., Järvinen, K.U., Jr., M.J.J., Chan, W.F., Huang, Z.: Provably Sublinear Point Multiplication on Koblitz Curves and its Hardware Implementation. IEEE Transaction on Computers 57(11), 1469–1481 (2008)
6. Järvinen, K., Skyttä, J.: On Parallelization of High-Speed Processors for Elliptic Curve Cryptography. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16(9), 1162–1175 (2008)
7. Kim, C.H., Kwon, S., Hong, C.P.: FPGA Implementation of High Performance Elliptic Curve Cryptographic Processor over $GF(2^{163})$. Journal of System Architcture 54(10), 893–900 (2008)
8. Massey, J., Omura, J.: Computational Method and Apparatus for Finite Arithmetic. US Patent (4587627), (1986)
9. Agnew, G.B., Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A.: An Implementation for a Fast Public-Key Cryptosystem. Journal of Cryptology 3(2), 63–79 (1991)
10. Kwon, S., Gaj, K., Kim, C.H., Hong, C.P.: "Efficient Linear Array for Multiplication in $GF(2^m)$ using a Normal Basis for Elliptic Curve Cryptography". In: Proceedings of CHES 2004, LNCS 3156, Springer-Verlag 76–91 (August 2004)
11. Wang, C.C., Truong, T.K., Shao, H.M., Deutsch, L.J., Omura, J.K., Reed, I.S.: VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$. IEEE Transaction on Computers 34(8), 709–717 (1985)
12. Gao, L., Sobelman, G.E.: "Improved VLSI Designs for Multiplication and Inversion in $GF(2^M)$ over normal bases". In: Proceedings of 13th Annual IEEE International ASIC/SOC Conference. 97–101 (2000)
13. Reyhani-Masoleh, A., Hasan, M.A.: A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$. IEEE Transactions on Computers 51(5), 511–520 (2002)
14. Ç. K. Koç, Sunar, B.: An Efficient Optimal Normal Basis Type II Multiplier over $GF(2^m)$. IEEE Transaction on Computers 50(1), 83–87 (2001)
15. Reyhani-Masoleh, A.: Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases. IEEE Transaction On Computers, 34–47 (2006)
16. Fan, H., Hasan, M.: Subquadratic computational complexity schemes for extended binary field multiplication using optimal normal bases. IEEE Transactions on Computers 56(10), 1435 (2007)

17. Gathen, J., Shokrollahi, A., Shokrollahi, J.: Efficient multiplication using type 2 optimal normal bases. In Carlet, C., Sunar, B., eds.: WAIFI. Volume 4547 of Lecture Notes in Computer Science., Springer 55–68 (2007)
18. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications. Cambridge University Press (1994)
19. Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A., Wilson, R.M.: Optimal Normal Bases in $GF(p^n)$. Discrete Appl. Math. 22(2), 149–161 (1989)
20. Gao, S., Lenstra, H.W.: Optimal Normal Bases. Designs, Codes and Cryptography 2, 315–323 (1992)
21. Ash, D.W., Blake, I.F., Vanstone, S.A.: Low Complexity Normal Bases. Discrete Applied Mathematics 25(3), 191–210 (1989)
22. Gustafsson, O., Olofsson, M.: Complexity reduction of constant matrix computations over the binary field. In: WAIFI. Volume 4547 of Lecture Notes in Computer Science., Springer 103–115 (2007)