Efficient and Low-complexity Hardware Architecture of Gaussian Normal Basis Multiplication over GF(2^m) for Elliptic Curve Cryptosystems

Bahram Rashidi[1], Sayed Masoud Sayedi[2], Reza Rezaeian Farashahi[3]
[1,2]Dept. of Elec. & Comp. Eng., Isfahan University of Technology, Isfahan 84156-83111, Iran
[3]Dept. of Mathematical Sciences, Isfahan University of Technology, Isfahan 84156-83111, Iran
[1]b.rashidi@ec.iut.ac.ir, [2]m_sayedi@cc.iut.ac.ir, [3]farashahi@cc.iut.ac.ir

***Abstract*—In this paper an efficient high-speed architecture of Gaussian normal basis multiplier over binary finite field GF(2^m) is presented. The structure is constructed by using regular modules for computation of exponentiation by powers of 2 and low-cost blocks for multiplication by normal elements of the binary field. Since the exponents are powers of 2, the modules are implemented by some simple cyclic shifts in the normal basis representation. As a result, the multiplier has a simple structure with a low critical path delay. The efficiency of the proposed structure is studied in terms of area and time complexity by using its implementation on Vertix-4 FPGA family and also its ASIC design in 180nm CMOS technology. Comparison results with other structures of the Gaussian normal basis multiplier verify that the proposed architecture has better performance in terms of speed and hardware utilization.**

Keywords: Finite Fields, Elliptic Curve Cryptosystems, Multiplication, Gaussian normal basis, FPGA, ASIC.

1. Introduction

Finite fields are applied in a variety of applications such as cryptography. The efficient implementations of finite fields are important in public key cryptosystems such as elliptic curve cryptosystem (ECC). In such cryptosystems, the multiplier is a key operator in group law and point multiplication [1]. Furthermore, the time and hardware complexity of multiplication are important factors in evaluating the efficiency of the related cryptosystems. The binary finite fields of order 2^m, denoted by GF(2^m), are attractive fields for implementation of ECC. In these fields, the addition operation is implemented by a simple bit-wise XOR. Moreover, the basis of a binary field is a critical factor in the hardware implementation. There are two popular and applicable basis called polynomial basis (PB) and normal basis (NB). In the normal basis representation, the squaring operation and every exponentiation by powers of 2 are implemented only by cyclic shift operations. This feature can be useful in the design of the field operations such as multiplier. Therefore, hardware implementation by normal basis representation is a notable issue in the cryptographic applications.

There are several presented architectures of the normal basis and Gaussian normal basis (GNB) multiplication in recent years [2]-[24]. For example, in [4] a novel scalable multiplication algorithm is presented for a Gaussian normal basis using Hankel Matrix-Vector representation. In [5] a modified digit-level GNB multiplier over GF(2^m) is proposed. Also for GNB of types greater than 2, a complexity reduction algorithm is proposed to reduce the number of XOR gates without increasing the gate delay of the digit-level multiplier. In [7] three structures for GNB multiplier are presented. The first structure is a low-complexity digit-level serial input parallel output (SIPO) GNB multiplier. Second structure is an improved digit-level parallel input serial output (PISO) multiplier architecture. And the third structure is a new hybrid architecture by connecting the output of the digit-level PISO multiplier to the input of the digit-level SIPO multiplier. In [8] a new normal basis multiplication algorithm based on divide-and-conquer and uniform shift method is used to implement an efficient multiplexer-based architecture. A bit-parallel GNB multiplier using one pipelined XOR tree is also designed in [15]. A novel algorithm for GNB binary finite field multiplication using Toeplitz matrix-vector representation is proposed in [19]. It is also shown that the GNB multiplication can be realized through block Toeplitz matrix-vector-products. The multipliers with systolic and semi-systolic architecture are presented in [3], [12], [14], [16], [17], [18] and [20]. A main problem in the systolic structure is its very high hardware consumption and high number of clock cycles; see for example [20] where the number of clock cycles is reduced. In [24] Dickson polynomial representation is proposed as an alternative way to represent the GNB of characteristic 2. A novel recursive Dickson–Karatsuba decomposition to achieve a subquadratic space-complexity parallel GNB multiplier is presented.

The aim of the present paper is to design a high-speed and efficient hardware architecture of the digit-serial Gaussian normal basis multiplier for binary finite fields. To that end, by reviewing the multiplication operation in the normal basis, we present a highly regular structure with low critical path delay and low hardware resources. The digit-serial multiplier is a suitable structure for area and speed trade-off in cryptographic application such as ECC. In addition, we present an efficient digit-serial multiplier based on exponentiation by powers of 2 and multiplication by a normal element of the binary finite field. Moreover, the proposed architecture is very regular and simple, and is well suited to hardware implementations. The FPGA and ASIC implementation results show that the proposed structure has acceptable area and time consumption.

The rest of this paper is organized as follows. In section 2, we briefly recall the notion of Gaussian normal basis for binary finite fields and propose the structure of digit-serial GNB multiplier. In section 3, we provide a comparison between this work and other previously related works. Finally, we conclude the paper in section 4.

2. Proposed structure of the Digit-serial Gaussian normal basis multiplier over GF(2^m)

A binary finite field of order 2^m denoted by GF(2^m) is isomorphic a vector space of dimension m over GF(2). So, the elements of GF(2^m) can be represented by a basis. Two important types of this representation in the finite field arithmetic are *polynomial basis* (PB) and *normal basis* (NB). For an efficient hardware implementation, the normal basis representation is a suitable choice. The element β in GF(2^m) is called a normal element if the set $\boldsymbol{B} = \{\beta^{2^0}, \beta^{2^1}, \beta^{2^2}, \dots, \beta^{2^{m-2}}, \beta^{2^{m-1}}\}$ is a basis for GF(2^m) over GF(2). For every binary finite field such a normal element exists and the corresponding set $\boldsymbol{B}$ is called a normal basis. Then, every element A of GF(2^m) is written by

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = \left(a_0\beta^{2^0} + a_1\beta^{2^1} + a_2\beta^{2^2} + \cdots + a_{m-2}\beta^{2^{m-2}} + a_{m-1}\beta^{2^{m-1}}\right),$$

where $a_i \in$ GF(2). For simplicity, the element A is represented by the m-bit number $[a_{m-1}, a_{m-2}, \dots, a_2, a_1, a_0]$. The addition of elements A, B given by $A = [a_{m-1}, a_{m-2}, \dots, a_2, a_1, a_0]$ and $B = [b_{m-1}, b_{m-2}, \dots, b_2, b_1, b_0]$ is

$$C = A + B = [a_{m-1} + b_{m-1}, a_{m-2} + b_{m-2}, \dots, a_2 + b_2, a_1 + b_1, a_0 + b_0].$$

So, the addition of the elements of GF(2^m) is performed using bit-wise XOR logic gates. The squaring of the element A is

$$A^2 = \left(\sum_{i=0}^{m-1} a_i \beta^{2^i}\right)^2 = \sum_{i=0}^{m-1} a_i^2 \left(\beta^{2^i}\right)^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = a_{m-1}\beta + \sum_{i=1}^{m-1} a_{i-1}\beta^{2^i},$$

This means A^2 is represented by $[a_{m-2}, a_{m-3}, \dots, a_2, a_1, a_0, a_{m-1}]$. So, one important property of using normal basis representation is performing the squaring operation very efficiently by a simple one-bit rotation to the left. Also, this operation is performed recursively for exponentiation of a power of two. Thus, for the positive integer n, the computation of A^{2^n} is performed via n-bit cyclic shift to left, i.e.,

$$A^{2^n} = [a_{m-n-1}, a_{m-n-2}, \dots, a_1, a_0, a_{m-1}, \dots, a_{m-n+1}, a_{m-n}].$$

And similarly $A^{2^{-n}}$ is computed by n-bit cyclic shift to right, as we have

$$A^{2^{-n}} = [a_{n-1}, a_{n-2}, \dots, a_1, a_0, a_{m-1}, a_{m-2}, \dots, a_{n+1}, a_n].$$

The multiplication of elements A, B in GF(2^m) is written by

$$C = AB = \sum_{i=0}^{m-1} a_i \beta^{2^i} \sum_{j=0}^{m-1} b_j \beta^{2^j} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \beta^{2^i + 2^j}$$

The element $\beta^{2^i+2^j}$ is represented by

$$\beta^{2^i+2^j} = \sum_{k=0}^{m-1} \lambda_{ij}^{(k)} \beta^{2^k}, \quad \lambda_{ij}^{(k)} \in \mathrm{GF}(2).$$

Thus,

$$C = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \sum_{k=0}^{m-1} \lambda_{ij}^{(k)} \beta^{2^k} = \sum_{k=0}^{m-1} c_k \beta^{2^k}, \qquad c_k = \sum_{i=0}^{m-1}\sum_{j-0}^{m-1} a_i b_j \lambda_{ij}^{(k)}.$$

Here $\lambda^{(k)}$ is an m by m symmetric matrix with entries $\lambda_{ij}^{(k)}$ in GF(2). The $\lambda^{(k)}$ matrices can be computed by a k-cyclic diagonal shift to $\lambda^{(0)}$ matrix; i.e., for the indices $i, j, k = 0, \dots, m-1$, computed modulo m, we have $\lambda_{i+1\, j+1}^{(k+1)} = \lambda_{ij}^{(k)}$. The matrix $\lambda^{(0)}$ is called the *multiplication matrix* and we recall it by M.

The complexity of the hardware implementation of a normal basis multiplication is related to the number of nonzero entries of the matrices M that is a crucial parameter for the speed of the system.
The *Gaussian normal basis*, GNB for short, is a special class of normal basis that by which the multiplication is simpler and more efficient [25] and [26]. The complexity of the multiplication of a GNB is measured by its type that is a positive integer related to the number of nonzero entries of the multiplication matrix. The time and area complexity of the multiplication operation over GF(2^m) is depend on the type of the normal basis with respect to that basis. Therefore, a more efficient multiplier has a smaller type. The *optimal normal basis* (ONB) is a GNB of type 1 or 2 providing the most efficient multiplication algorithm among all normal bases. The GNB is considered in several standards such as IEEE P1363 [26] and NIST [27]. For example even types T={4,2,6,4, and 10} corresponded to fields {GF(2^{163}), GF(2^{233}), GF(2^{283}), GF(2^{409}) and GF(2^{571})}, are recommended by these two standards.
For each binary finite field GF(2^m), where m is not divisible by 8, a GNB exists of some type, also for each positive integer T at most one GNB of type T exists. More precisely, for given positive integers m and T, let $p = mT + 1$ be a prime number such that $\gcd(mT/k, m) = 1$, where k is the multiplicative order of 2 module p. Then a GNB over GF(2^m) of type T exists. In this work, we consider the GNBs with odd values of m which are applicable for cryptography applications and implies that T is an even number.
The GNB multiplication can also be computed via the following approach; see e.g. [26]. Let GF(2^m) has a Gaussian normal basis $\boldsymbol{B}$ of type T, where $p = mT + 1$ is a prime number. Let u be an integer of order T mod p. Then, the set

$$Z = \left\{ z_{i,j} : \ z_{i,j} = 2^i u^j \ \mid \ i \in \{0,1,\dots,m-1\}, \qquad j \in \{0,1,\dots,T-1\} \right\}$$

is a reduced residue system modulo p, i.e., each positive integer x less than p can be uniquely represented as $x = z_{i,j} \bmod p$. Let F be a function given by

$$\begin{gathered} F : \{1,2,\dots,p-1\} \to \{0,1,\dots,m-1\} \\ F(x) = i, \quad x = z_{i,j} \bmod p \end{gathered} \qquad (1)$$

For even type T the first coordinate c_0 of C, the multiplication A and B, is computed by

$$c_0 = \sum_{k=1}^{p-2} a_{F(k+1)} b_{F(p-k)}. \qquad (2)$$

Also, other coordinates c_i, $1 \le i \le m-1$, are computed similarly by one bit right cyclic shift of inputs.
A bit serial implementation of normal basis multiplication of two elements A and B is performed in [21] as follows.

$$\begin{aligned} C \ &= \ AB = A\left(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots + b_0\beta\right) \\ &= b_{m-1}\left(A\beta^{2^{m-1}}\right) + b_{m-2}\left(A\beta^{2^{m-2}}\right) + \cdots + b_0(A\beta) \\ &= b_{m-1}\left(A^{2^{-(m-1)}}\beta\right)^{2^{m-1}} + b_{m-2}\left(A^{2^{-(m-2)}}\beta\right)^{2^{m-2}} + \cdots + b_0(A\beta) \\ &= \left(\dots\left(\left(b_{m-1}A^{2^{-(m-1)}}\beta\right)^2 + b_{m-2}A^{2^{-(m-2)}}\beta\right)^2 \dots\right)^2 + b_0(A\beta). \end{aligned}$$

Also the digit-serial GNB multiplication is implemented as below, where B is divided into d words of w bits with $d = \left\lceil \frac{m}{w} \right\rceil$.

$$\begin{aligned} C = &\left(\dots\left(\left(b_{m-1}A^{2^{-(w-1)}}\beta^{2^{m-w}}\right)^2 + b_{m-2}A^{2^{-(w-2)}}\beta^{2^{m-w}}\right)^2 + \cdots\right)^2 + b_{m-w}A\beta^{2^{m-w}} \\ &+ \left(\dots\left(\left(b_{m-w-1}A^{2^{-(w-1)}}\beta^{2^{m-2w}}\right)^2 + b_{m-w-2}A^{2^{-(w-2)}}\beta^{2^{m-2w}}\right)^2 + \cdots\right)^2 + b_{m-2w}A\beta^{2^{m-2w}} \\ &+ \cdots \\ &+ \left(\dots\left(\left(b_{m-(d-1)w-1}A^{2^{-(w-1)}}\beta^{2^{m-dw}}\right)^2 + b_{m-(d-1)w-2}A^{2^{-(w-2)}}\beta^{2^{m-dw}}\right)^2 + \cdots\right)^2 \\ &+ b_0A\beta. \end{aligned}$$

In this method multiplications by some powers of β are required which cause complex and irregular structure in hardware implementation [21]. To have a low-complexity and regular architecture of multiplication by $\beta^{2^{(m-iw)}}$ for some integer i, the computation $y = x\beta^{2^{(m-iw)}}$ can be performed in three steps; first the exponentiation of the input x by $2^{-(m-iw)}$ is done, then multiplication by β is performed, and finally the exponentiation of the result by $2^{(m-iw)}$ is completed. These operations result in:

$$y = x\beta^{2^{(m-iw)}} = \left(\left(x^{2^{-(m-iw)}}\right)\beta\right)^{2^{(m-iw)}}.$$

Fig. 1 shows the proposed method for multiplication by $\beta^{2^{(m-iw)}}$.

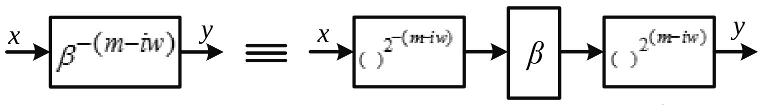


Fig. 1: Proposed implementation for multiplication by $\beta^{2^{(m-iw)}}$

In the proposed method two steps of exponentiation by $2^{-(m-iw)}$ and $2^{(m-iw)}$ are free hardware implemented only by cyclic shift; therefore, only multiplication by β is the main part to be implemented.
Here, we explain the structure of the proposed digit-serial multiplier. Let A, B be two elements in GF(2^m) and let $B = [b_{m-1}, b_{m-2}, \dots, b_2, b_1, b_0]$. We consider B as the following $w \times d$ array and divide it into its columns.

$$\begin{pmatrix} b_{m-1} & b_{m-2} & \cdots & b_{m-w} \\ b_{m-w-1} & b_{m-w-2} & & b_{m-2w} \\ & \vdots & \ddots & \vdots \\ b_{m-(d-1)w-1} & b_{m-(d-1)w-2} & \cdots & b_{m-dw} \end{pmatrix}$$

We let $b_i = 0$ if $i \leq 0$. More precisely,

$$B = B_1 + B_2 + B_3 + \cdots + B_w,$$

where, for $i = 1, \dots, w$,

$$B_i = \sum_{k=1}^{d} b_{m-(k-1)w-i}\, \beta^{2^{m-(k-1)w-i}}.$$

The multiplication of elements A, B in GF(2^m) is written by

$$C = AB = A\sum_{i=1}^{w} B_i = A\sum_{i=1}^{w}\sum_{k=1}^{d} b_{m-(k-1)w-i}\, \beta^{2^{m-(k-1)w-i}} = \sum_{i=1}^{w}\sum_{k=1}^{d} b_{m-(k-1)w-i}\, A\beta^{2^{m-(k-1)w-i}}$$
$$= \sum_{i=1}^{w}\left(\sum_{k=1}^{d} b_{m-(k-1)w-i}\, A^{2^{-(w-i)}}\beta^{2^{m-kw}}\right)^{2^{w-i}}.$$

In other words, we have

$$C = \sum_{i=1}^{w} C_i^{2^{w-i}} = \left(\left(\dots\left(\left(C_1^{\ 2} + C_2\right)^2 + C_3\right)^2 + \cdots\right)^2 + C_w\right),$$

where for $i = 1, \dots, w$,

$$C_i = \sum_{k=1}^{d} b_{m-(k-1)w-i}\, A^{2^{-(w-i)}}\beta^{2^{m-kw}} = \sum_{k=1}^{d} b_{m-(k-1)w-i}\left(\left(\left(A^{2^{-(w-1)}}\right)^{2^{(i-1)}}\right)^{2^{-(m-kw)}}\beta\right)^{2^{m-kw}}.$$

To calculate exponentiation by $2^{-(m-kw)}$ before multiplication by β block in regular form, we write

$$\left(\left(A^{2^{-(w-1)}}\right)^{2^{(i-1)}}\right)^{2^{-(m-kw)}} = \left(\dots\left(\left(\left(A^{2^{-(w-1)}}\right)^{2^{(i-1)}}\right)^{2^{-(m-w)}}\right)^{2^w}\dots\right)^{2^w}.$$

So, first exponentiation by $2^{-(m-w)}$ is computed, and then for $k = 2,3,\dots,d$, exponentiation by $2^{-(m-kw)}$ are generated by a sequence of exponentiation by 2^w with length d-1. Fig.2 shows the proposed structure for the digit-serial GNB multiplier over GF(2^m).

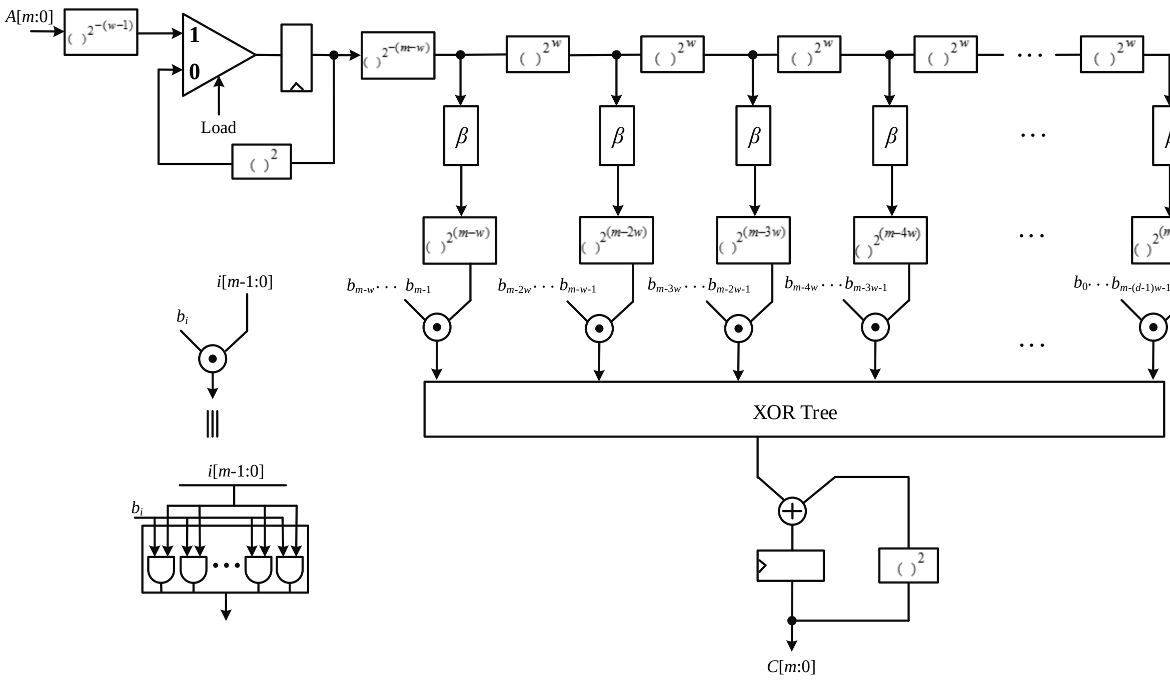


Fig. 2: Proposed structure for the digit-serial GNB multiplier over GF(2^m)

As seen in the Fig. 2, a regular architecture for hardware implementation is provided. In proposed structure different exponentiation by power of 2 blocks are implemented by wired cyclic shift in the normal basis. This property is an important factor for improvement of efficiency in the structure.

For implementation of multiplication by β in GF(2^m) of type T, a method presented in [6] is employed. In this method, the entries of the multiplication matrix M are encoded to an $(m-1)\times T$ matrix R with entries $r_{i,j} \in \{0,1,2,\dots,m-1\}$, where $i = 0,1,\dots,m-2$ and $j = 0,1,\dots,T-1$. Notice that the multiplication matrix M is symmetric and for GNB of even type T, we have $\lambda_{i,0}^{(k)} = \lambda_{i,k}^{(0)}$, where i,k are in $\{0,1,2,\dots,m-1\}$ (see [6]). So,

$$\beta^{2^i}\beta = \sum_{k=0}^{m-1}\lambda_{i,0}^{(k)}\beta^{2^k} = \sum_{k=0}^{m-1}\lambda_{i,k}^{(0)}\beta^{2^k}.$$

This means, the ith row of matrix M is the m-bit number representation of $\beta^{2^i}\beta$. Thus, the number of entries '1' in the first row is one and in other rows is even and less or equal to T. Now, the entries of ith row of matrix R are identified based on the column numbers of entries 1 in $(i+1)$th row of matrix M. If the number of ones in row $(i+1)$ of M matrix is T, then all entries of row i of matrix R are specified. If it is not the case, the remaining entries of R, whose number is even, is initialized with a constant value. There is also another method to determine matrix R which is based on the function F in Eq.(1). In this method, for all $k = 1,2,\dots,p-2$, where $p = mT+1$, the pairs $(F(k+1), F(p-k))$ are calculated, that for each calculated pair its second coordinate, the value $F(p-k)$, is an entry of matrix R in row $F(k+1)-1$ if $F(k+1)\geq 1$. Multiplication by β based on matrix R is as follows

$$\beta B = \beta\sum_{i=0}^{m-1} b_i\beta^{2^i} = \sum_{i=0}^{m-1} b_i\beta\beta^{2^i} = \sum_{i=0}^{m-1} b_i\sum_{k=0}^{m-1}\lambda_{i,0}^{(k)}\beta^{2^k} = \sum_{i=0}^{m-1}\sum_{k=0}^{m-1} b_i\lambda_{i,k}^{(0)}\beta^{2^k} = \sum_{k=0}^{m-1}\sum_{i=0}^{m-1} b_i\,\lambda_{i,k}^{(0)}\beta^{2^k}$$
$$= \sum_{k=0}^{m-1}\left(\sum_{i=0}^{m-1} b_i\,\lambda_{k,i}^{(0)}\right)\beta^{2^k} = \sum_{k=0}^{m-1}\sum_{j=0}^{T-1} b_{r_{k,j}}\beta^{2^k} = \sum_{k=0}^{m-1} s(k,B)\beta^{2^k},$$

where, $s(k,B) = \sum_{j=0}^{T-1} b_{r_{k,j}}$, $0 \leq k \leq m-2$. Briefly, we have

$$\beta B = [s(m-1,B),\dots,s(2,B),s(1,B),b_1].$$

In the following an example of GF(2^7), with type T=4 GNB is presented. For the following multiplication matrix M,

$$M=\begin{bmatrix}0&1&0&0&0&0&0\\1&0&1&0&0&1&1\\0&1&0&1&1&1&0\\0&0&1&0&0&1&0\\0&0&1&0&0&0&1\\0&1&1&1&0&0&1\\0&1&0&0&1&1&1\end{bmatrix},$$

by using the first method the matrix R is calculated as follows:

$$R=\begin{bmatrix}0&2&5&6\\1&3&4&5\\2&5&3&3\\2&6&0&0\\1&2&3&6\\1&4&5&6\end{bmatrix}.$$

Also based on the second method, first values of $F(k)$ are calculated, as shown in Table 2.

Table2: Sequence of F for T=4 over GF(2^7)

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$F(k)$	0	1	5	2	1	6	5	3	3	2	4	0	4	6
k	15	16	17	18	19	20	21	22	23	24	25	26	27	28
$F(k)$	6	4	0	4	2	3	3	5	6	1	2	5	1	0

Then the pairs of $(F(k+1), F(p-k))$ are listed as shown in Table 3, and based on that matrix R can be constructed.

Table3: Pairs of $(F(k+1), F(p-k))$ for T=4 over GF(2^7)

(1,0)	(4,2)	(3,2)
(5,1)	(0,4)	(3,3)
(2,5)	(4,0)	(5,3)
(1,2)	(6,4)	(6,5)
(6,1)	(6,6)	(1,6)
(5,6)	(4,6)	(2,1)
(3,5)	(0,4)	(5,2)
(3,3)	(4,0)	(1,5)
(2,3)	(2,4)	(0,1)

If we show output bits of multiplication βB by $t_k = s(k, B)$ then for above matrix R we have,

$t_0 = b_1$
$t_1 = s(1, B) = b_0 \oplus b_2 \oplus b_6 \oplus b_5$
$t_2 = s(2, B) = b_5 \oplus b_3 \oplus b_4 \oplus b_1$
$t_3 = s(3, B) = b_5 \oplus b_3 \oplus b_2 \oplus b_3 = b_5 \oplus b_2$
$t_4 = s(4, B) = b_2 \oplus b_0 \oplus b_6 \oplus b_0 = b_2 \oplus b_6$
$t_5 = s(5, B) = b_1 \oplus b_6 \oplus b_3 \oplus b_2$
$t_6 = s(6, B) = b_1 \oplus b_4 \oplus b_6 \oplus b_5$

As it can be seen, there are some common XOR terms in t_k expressions. For example t_1 and t_6 have a common term $b_6 \oplus b_5$, and t_4 and t_5 have a common term $b_2 \oplus b_6$. Considering these common terms, hardware implementation of the multiplication by β over GF(2^7) is as shown in Fig. 3.

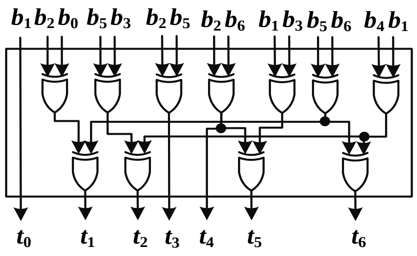
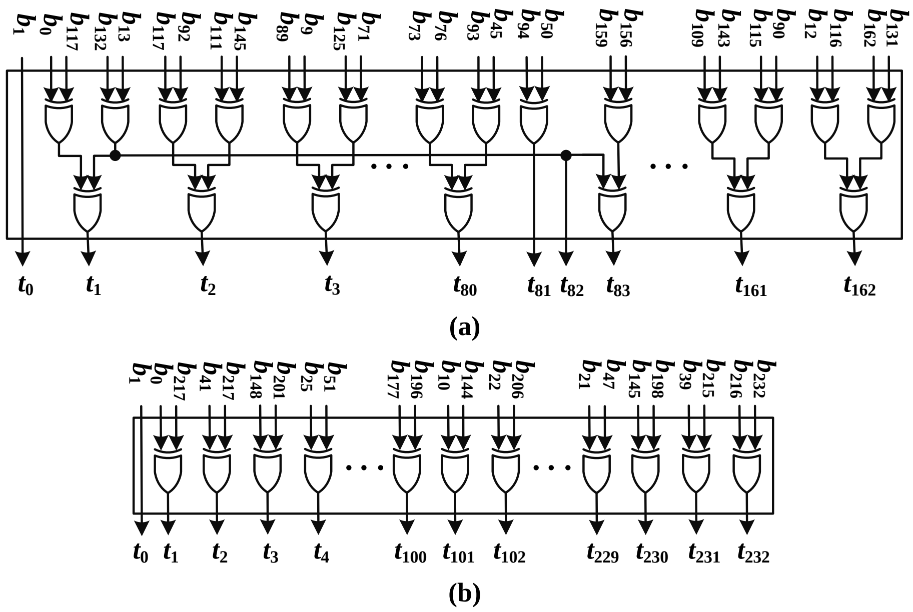

$b_1 b_2 b_0$ $b_5 b_3$ $b_2 b_5$ $b_2 b_6$ $b_1 b_3 b_5 b_6$ $b_4 b_1$
t_0 t_1 t_2 t_3 t_4 t_5 t_6
Fig. 3: Hardware implementation of the multiplication by β over GF(2^7)

Fig.4 (a) and Fig.4 (b) show the hardware implementation of the multiplication by β over two applicable fields GF(2^{163}) and GF(2^{233}) respectively.


b_1 b_0 b_{117} b_{132} b_{13} b_{117} b_{92} b_{111} b_{145} b_{89} b_9 b_{125} b_{71} b_{73} b_{76} b_{93} b_{45} b_{94} b_{50} b_{159} b_{156} b_{109} b_{143} b_{115} b_{90} b_{12} b_{116} b_{162} b_{131}
t_0 t_1 t_2 t_3 t_{80} t_{81} t_{82} t_{83} t_{161} t_{162}
(a)
b_1 b_0 b_{217} b_{41} b_{217} b_{148} b_{201} b_{25} b_{51} b_{177} b_{196} b_{10} b_{144} b_{22} b_{206} b_{21} b_{47} b_{145} b_{198} b_{39} b_{215} b_{216} b_{232}
t_0 t_1 t_2 t_3 t_4 t_{100} t_{101} t_{102} t_{229} t_{230} t_{231} t_{232}
(b)
Fig.4: Hardware implementation of the multiplication by β for fields GF(2^{163}) (a) and GF(2^{233}) (b).

Type of the Gaussian normal basis in the field GF(2^{163}) is T=4. In this field, after resource sharing of the XOR common terms 24% of XOR gates are reduced in the implementation.

In the following, the proposed structure of digit-serial GNB multiplier is presented for two cases of w=3, $d=\left\lceil\frac{7}{3}\right\rceil$=3 and w=4, $d=\left\lceil\frac{7}{4}\right\rceil$=2 over GF($2^7$). For the first case, B is represented by three words B_1 to B_3:

$$\begin{aligned}B_1 &= b_6\beta^{2^6}+b_5\beta^{2^5}+b_4\beta^{2^4}\\ B_2 &= b_3\beta^{2^3}+b_2\beta^{2^2}+b_1\beta^{2^1}\\ B_3 &= b_0\beta\end{aligned}$$

And for $i=1,2,3,\ C_i$ are:

$$C_1=\left(\left(A^{2^{-2}}\right)^{2^{-4}}\beta\right)^{2^4}b_6+\left(\left(\left(A^{2^{-2}}\right)^{2^{-4}}\right)^{2^3}\beta\right)^2 b_3+\left(\left(\left(\left(A^{2^{-2}}\right)^{2^{-4}}\right)^{2^3}\right)^{2^3}\beta\right)^{2^{-2}}b_0$$

$$C_2=\left(\left(\left(A^{2^{-2}}\right)^2\right)^{2^{-4}}\beta\right)^{2^4}b_5+\left(\left(\left(\left(A^{2^{-2}}\right)^2\right)^{2^{-4}}\right)^{2^3}\beta\right)^2 b_2+\left(\left(\left(\left(\left(A^{2^{-2}}\right)^2\right)^{2^{-4}}\right)^{2^3}\right)^{2^3}\beta\right)^{2^{-2}}b_{-1}$$

$$C_3=\left(\left(\left(A^{2^{-2}}\right)^{2^2}\right)^{2^{-4}}\beta\right)^{2^4}b_4+\left(\left(\left(\left(A^{2^{-2}}\right)^{2^2}\right)^{2^{-4}}\right)^{2^3}\beta\right)^2 b_1+\left(\left(\left(\left(\left(A^{2^{-2}}\right)^{2^2}\right)^{2^{-4}}\right)^{2^3}\right)^{2^3}\beta\right)^{2^{-2}}b_{-2}.$$

The bits b_{-1} and b_{-2} are set to zero, and product C of the Gaussian normal basis multiplication based on C_1, C_2 and C_3 is presented as follows:

$$C=\left(\left({C_1}^2+C_2\right)^2+C_3\right).$$

Table 3 shows required exponentiation operations in the proposed digit-serial GNB multiplier over GF(2^7) for w=3 and d=3, and Fig. 5 shows the proposed structure of the digit-serial GNB multiplier over GF(2^7) with T=4 for case w=3 and d=3.

Table 3: Exponentiation operations in the proposed digit-serial GNB multiplier over GF(2^7) for w=3 and d=3

Exponentiation operations	**w=3, d=3**	**Vector representation**
$()^{2^{-(w-1)}}$	$()^{2^{-2}}$	$(a_1, a_0, a_6, a_5, a_4, a_3, a_2)$
$()^{2^{-(m-w)}}$	$()^{2^{-4}}$	$(a_3, a_2, a_1, a_0, a_6, a_5, a_4)$
$()^{2^{w}}$	$()^{2^{3}}$	$(a_3, a_2, a_1, a_0, a_6, a_5, a_4)$
$()^{2^{(m-w)}}$	$()^{2^{4}}$	$(a_2, a_1, a_0, a_6, a_5, a_4, a_3)$
$()^{2^{(m-2w)}}$	$()^{2^{1}}$	$(a_5, a_4, a_3, a_2, a_1, a_0, a_6)$
$()^{2^{(m-3w)}}$	$()^{2^{-2}}$	$(a_1, a_0, a_6, a_5, a_4, a_3, a_2)$

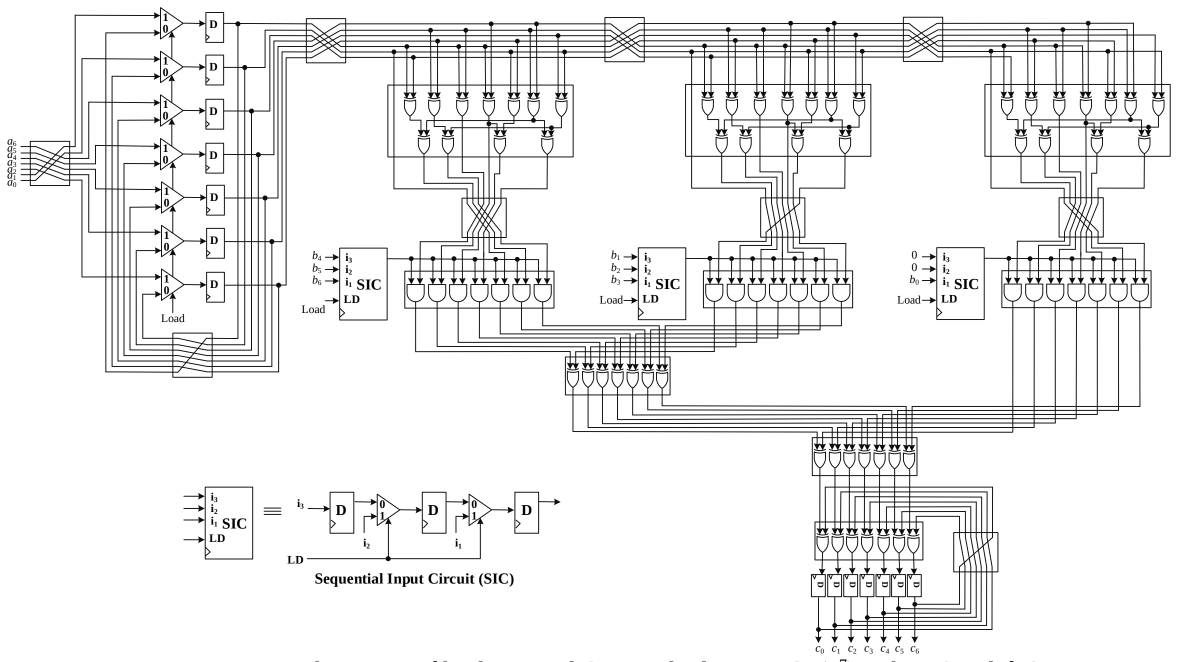


Fig. 5: Proposed structure ofthe digit-serial GNB multiplier over GF(2^7) with w=3 and d=3

For the case of w=4 and d=2 the words representation of the B is

$$B_1 = b_6\beta^{2^6} + b_5\beta^{2^5} + b_4\beta^{2^4} + b_3\beta^{2^3}$$
$$B_2 = b_2\beta^{2^2} + b_1\beta^{2^1} + b_0\beta$$

and multiplication result is $C = \left(\left(\left(C_1{}^2 + C_2\right)^2 + C_3\right)^2 + C_4\right)$ where C_1 and C_4 are:

$$C_1 = \left(\left(A^{2^{-3}}\right)^{2^{-3}}\beta\right)^{2^3} b_6 + \left(\left(\left(A^{2^{-3}}\right)^{2^{-3}}\right)^{2^4}\beta\right)^{2^{-1}} b_2$$

$$C_2 = \left(\left(\left(A^{2^{-3}}\right)^{2}\right)^{2^{-3}}\beta\right)^{2^3} b_5 + \left(\left(\left(\left(A^{2^{-3}}\right)^{2^2}\right)^{2^{-3}}\right)^{2^4}\beta\right)^{2^{-1}} b_1$$

$$C_3 = \left(\left(\left(A^{2^{-3}}\right)^{2^2}\right)^{2^{-3}}\beta\right)^{2^3} b_4 + \left(\left(\left(\left(A^{2^{-3}}\right)^{2^2}\right)^{2^{-3}}\right)^{2^4}\beta\right)^{2^{-1}} b_0$$

$$C_4 = \left(\left(\left(A^{2^{-3}}\right)^{2^3}\right)^{2^{-3}}\beta\right)^{2^3} b_3 + \left(\left(\left(\left(A^{2^{-3}}\right)^{2^3}\right)^{2^{-3}}\right)^{2^4}\beta\right)^{2^{-1}} b_{-1}.$$

Required exponentiation operations in the proposed digit-serial GNB multiplier over GF(2^7) for w=4 and d=2 are shown in Table 4, and the proposed structure of the digit-serial GNB multiplier over GF(2^7) with w=4 and d=2 is shown Fig. 6.

Table 4: Exponentiation operations in proposed digit-serial GNB multiplier over GF(2^7) for w=4 and d=2

Exponentiation operations	w=4, d=2	Vector representation
$()^{2^{-(w-1)}}$	$()^{2^{-3}}$	$(a_2, a_1, a_0, a_6, a_5, a_4, a_3)$
$()^{2^{-(m-w)}}$	$()^{2^{-3}}$	$(a_2, a_1, a_0, a_6, a_5, a_4, a_3)$
$()^{2^{w}}$	$()^{2^{4}}$	$(a_2, a_1, a_0, a_6, a_5, a_4, a_3)$
$()^{2^{(m-w)}}$	$()^{2^{3}}$	$(a_3, a_2, a_1, a_0, a_6, a_5, a_4)$
$()^{2^{(m-2w)}}$	$()^{2^{-1}}$	$(a_0, a_6, a_5, a_4, a_3, a_2, a_1)$

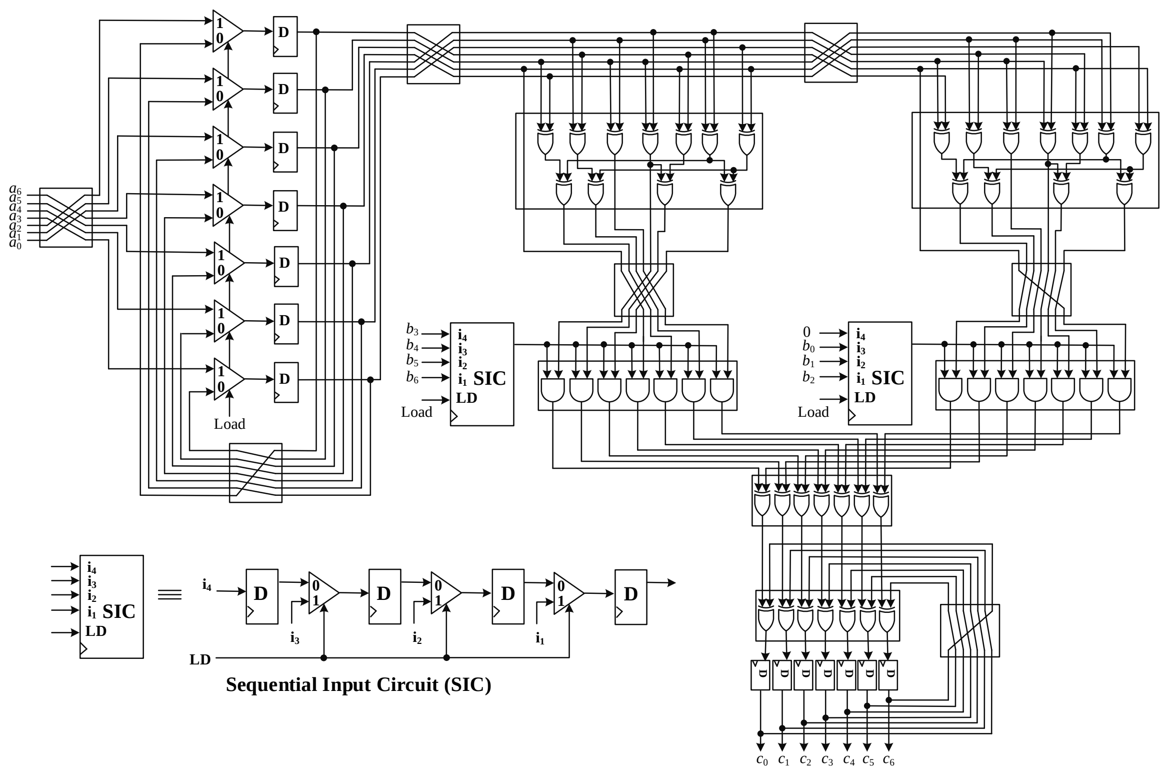


Fig. 6: Proposed structure of the digit-serial GNB multiplier over GF(2^7) with w=4 and d=2

The critical data path of the proposed structure of digit-serial GNB multiplier over GF(2^m) with type T is T_A+ $(\lceil log_2^T \rceil + \lceil log_2^{(d+1)} \rceil)T_X$, where T_A and T_X denote the time delay of a 2-input AND gate and 2-input XOR gate respectively. For above two examples the critical data path of structures in Fig.5 and Fig.6 are $T_A + (\lceil log_2^4 \rceil + \lceil log_2^{(3+1)} \rceil)T_X = T_A + 5T_X$ and $T_A + (\lceil log_2^4 \rceil + \lceil log_2^{(2+1)} \rceil)T_X = T_A + 4T_X$ respectively. Also, the number of clock cycles for the case (w=3, d=3) is 4 and for case (w=4, d=2) is 5. The proposed digit-serial GNB multiplier requires $d.m$ AND gates and $\leq d.m$+(T-1)(m-1) XOR gates. The number of XOR gates is lower than those of other digit-serial structures. More details of the hardware and time complexity of this work and other related works are presented in the next section.

3. Comparison Results

Comparison with other structures of the GNB and ONB multipliers based on different parameters like hardware resources, critical path delay and number of clock cycles are presented here. The hardware implementation of the proposed architecture is based on two fields of GF(2^{163}) and GF(2^{233}) recommended by NIST for ECC applications. The proposed digit-serial GNB multiplier structure has been successfully verified and implemented using Xilinx ISE 11onVirtex-4XC4VLX100-ff1148FPGA. In addition, the ASIC results are achieved by using Synopsys Design Vision tool based on library of standard cells with 180nm CMOS technology. In Table 5, hardware utilization including numbers of 2-input XOR gates, 2-input AND gates, D flip-flops and 2 to 1 multiplexers for proposed structure and also for previously related works are presented. Also critical path delay and latency of multipliers are presented.

Table 5: Comparison of the proposed digit-serial GNB multiplier and other related works

Works \ Structures	**# 2-input XOR**	**# 2-input AND**	**# FF**	**# 2-to-1 Mux**	**Critical path delay**	**Latency (cycle)**
[2] Bit-Parallel ONB T=2	m^2+2m-1	m	---	$\frac{m(m-1)}{2}$	$T_M + [2 + log_2^{(m-1)}]T_X$	---
[3] Bit-Parallel GNB	$m^2+2mT+1$	m^2	$2m^2+mT+1$	$mT+1$	$T_A+T_X+T_L$	$m+(T+1)^2-1$
[4] Digit-serial GNB	$d^2+d+mT+1$	d^2	$3.5d^2+5(mT+1)+3nd$	$mT+1$	T_A+T_X	$d+n(n+1)$
[5] Digit-serial GNB	$\leq\frac{d(m-1)}{2}(T-1)+dm$	dm	$3m$	---	$T_A + [\lceil log_2^T\rceil + \lceil log_2^{(d+1)}\rceil]T_X$	$\lceil\frac{m}{d}\rceil + 1$
[6] Digit-serial GNB DLGMs	$\leq d((m-\frac{(d+1)}{2})T+\frac{(d+1)}{2})$	dm	$2m$	---	$T_A + [\lceil log_2^T\rceil + \lceil log_2^{(m)}\rceil]T_X$	$\lceil\frac{m}{d}\rceil +1$
[6] Digit-serial GNB DLGMp	$\leq\frac{d}{2}(Tm-T+m+1)$	dm	$3m$	---	$T_A + [\lceil log_2^T\rceil + \lceil log_2^{(d+1)}\rceil]T_X$	$\lceil\frac{m}{d}\rceil +1$
[7] Digit-serial GNB	$\leq (d(m-1)-\frac{d(d-1)}{2})(T-1)+dm$	dm	$2m$	$2m$	$T_A + [\lceil log_2^T\rceil + \lceil log_2^{(d+1)}\rceil]T_X$	$\lceil\frac{m}{d}\rceil +1$
[8] Bit-Parallel ONB T=2	$(3m^2-m+6)/4$	---	---	$m(m-1)/2$	$T_M + [2 + log_2^{(m-1)}]T_X$	1
[8] Bit-Parallel ONB T=1	$(m^2+3m-2)/2$	---	---	$m(m-1)/2$	$T_M + [2 + log_2^{(m-1)}]T_X$	1
[9] Bit-Parallel GNB	$m^2T^2 + 2mT$	$(mT+ 1)^2$	---	---	$T_A + \left[\lceil log_2^{(mT+1)}\rceil T_X\right] + T_X$	1
[22] Bit-Parallel GNB	$mT\lceil log_2^T\rceil + m^2$	m^2	---	---	$T_A + \left[m + \lceil log_2^{(T)}\rceil\right] T_X$	1
[10] Digit-serial ONB T=2 AEDS	$w(3m-w-2)$	$w(m-0.5w+0.5)$	---	---	$T_A + \left[1 + \lceil log_2^{(m)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[10] Digit-serial ONB T=2 XEDS	$w(2m-0.5w-1.5)$	$w(2m-w)$	---	---	$T_A + \left[1 + \lceil log_2^{(m)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[10] Digit-serial ONB T=1 AEDS	$(w+1)(1.5m-2)+1$	$w(m-1)+m/2$	---	---	$T_A + \left[1 + \lceil log_2^{(m)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[10] Digit-serial ONB T=1 XEDS	$(w+1)(m-1)$	$w(m-1)+m$	---	---	$T_A + \left[1 + \lceil log_2^{(m)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[11] Bit-Parallel ONB T=2	$3/2m(m-1)$	m^2	---	---	$T_A + \left[1 + \lceil log_2^{(m)}\rceil\right] T_X$	---
[12] Bit-serial ONB T=1	$2m+1$	$m+1$	$4m+3$	---	$T_A+T_X+T_L$	$m(m+1)$
[13] Digit-serial ONB T=2 w-SMPO$_I$	$w(2m-1)$	$w(m/2+1)+m$	$3m$	---	$\leq 2T_A + \left[3 + \lceil log_2^{(w-1)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[13] Digit-serial ONB T=2 w-SMPO$_{II}$	$w(m+m/2)$	$wm+m$	$3m$	---	$\leq 2T_A + \left[3 + \lceil log_2^{(w-1)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[13] Digit-serial ONB T=1 w-SMPO$_I$	$3wm/2+m+w+1$	$wm/2+m+w+1$	$3m$	---	$\leq 2T_A + \left[3 + \lceil log_2^{(w-1)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[13] Digit-serial ONB T=1 w-SMPO$_{II}$	$wm+w+m+1$	$wm+w+m+1$	$3m$	---	$\leq 2T_A + \left[3 + \lceil log_2^{(w-1)}\rceil\right] T_X$	$\lceil\frac{m}{d}\rceil +1$
[14]Bit-Parallel GNB	$(mT)^2 / 4$	$(mT)^2 / 4$	$9(mT)^2/8+9mT/4$	---	$T_A+T_X+T_L$	$mT/2 + 1$
[15]Bit-Parallel GNB	mT	mT	$2mT$	---	$T_A + \lceil log_2^{(mT)}\rceil (T_X + T_L)$	$\lceil log_2^{(mT+1)}\rceil + mT$
[16] Bit-Parallel ONB T=2	m^2(3-input XOR)	$2m^2$	$3m^2$	---	$T_A+T_{3X}+T_L$	m
[17] Bit-Parallel GNB	$(mT)^2/2+5mT/2+1$	$(mT)^2/2+mT/2$	$9(mT)^2/8+5mT/4+2$	---	$T_A+T_X+T_L$	$mT/2+1$
[18] Bit-Parallel ONB T=2	$2m^2+2m$ (3-input XOR)	$2m^2+m$	$5m^2+2m-2$	---	$T_A+T_{3X}+T_L$	$m+1$
[19] Digit-serial GNB	kd^2	kd^2	$k(2d+1)$	k	$T_A + [1 + \lceil log_2^d\rceil]T_X$	$\lceil\frac{mT+1}{d}\rceil+k-1$
[21] Digit-serial ONB T=2	$d(2m-1)$	dm	$3m$	---	$T_A + [1 + \lceil log_2^{(d+1)}\rceil]T_X$	$\lceil\frac{m}{d}\rceil +1$
[20] Digit-serial GNB	$\leq\frac{\lceil\sqrt{\frac{m}{d}}\rceil d(m-1)}{2}(T-1) + \left(1 + \lceil\sqrt{\frac{m}{d}}\rceil d\right)m$	$\lceil\sqrt{\frac{m}{d}}\rceil dm$	$(1+3\lceil\sqrt{\frac{m}{d}}\rceil)m$	---	$T_A + [\lceil log_2^T\rceil + \lceil log_2^{(d+1)}\rceil]T_X$	$\leq 2\lceil\sqrt{\frac{m}{d}}\rceil$
[23] Bit-Parallel GNB	$(\lceil\frac{mT}{4}\rceil+1)mT+\lceil\frac{mT}{4}\rceil-2$	$\frac{mTf}{2}\lceil\frac{mT}{4}\rceil$	$2mT(f+1)+\frac{(f+1)(f+2)}{2}$	f	$T_A+T_X+T_L$	$f+2$
[24] Bit-Parallel GNB, b=2 T=4	$36m^{log_2^3}-28m+2$	$3m^{log_2^3}$	---	---	$T_A + [6 + 5\lceil log_2^m\rceil]T_X$	---
[24] Bit-Parallel GNB, b=3 T=4	$40.88m^{log_3^6}-31m+2.8$	$3.1m^{log_3^6}$	---	---	$T_A + [6 + 8\lceil log_3^m\rceil]T_X$	---
Proposed method Digit-serial GNB	$\boldsymbol{\leq (m-1)(T-1)+dm}$	$\boldsymbol{dm}$	$\boldsymbol{3m}$	$\boldsymbol{2m}$	$\boldsymbol{T_A + [\lceil log_2^T\rceil + \lceil log_2^{(d+1)}\rceil]T_X}$	$\boldsymbol{\lceil\frac{m}{d}\rceil +1}$

Note: d: digit size; $w=\lceil\frac{m}{d}\rceil$: number of words; $f=\lceil\frac{mT}{4}\rceil$; $n = \lceil\frac{mT+1}{d}\rceil$;$T_A$, T_X, T_L, T_M and T_{3X} denote time delay of a 2-input AND gate, 2-input XOR gate, one bit Latch, and 2 to 1 multiplexer and 3-input XOR gate respectively; $q = dk$ is consecutive coordinate to satisfy the corresponding normal basis representation, where $q \geq mT/2$ if m is even, and $q \geq (mT-T+2)/2$ if m is odd; b: $m = b^i$ (i>1).

As seen in the Table 5 hardware utilization in the proposed structures is comparable with other digit-serial GNB multipliers. The proposed digit-serial GNB multiplier requires dm AND gates and $\leq dm+(T-1)(m-1)$ XOR gates.

The number of XOR gates is the lowest compared to other digit-serial structures. Hardware resources in recent work [20] are more than that of present design; however, the latency in [20] is better. Table 6 shows FPGA implementation results of the proposed architecture and work [7] on Virtex-4 XC4VLX100-ff1148for GF(2^{233}) and GF(2^{163}). In the table hardware utilization, maximum frequency and execution time for different digit sizes are reported.

Table 6: FPGA implementation results of the proposed architectures for GF(2^{233}) and GF(2^{163})on Virtex-4 XC4VLX100-ff1148

Works and Fields	Digit size	Area	Fmax(MHz)	Time (ns)
[7] DL-SIPO GF(2^{163})	41	7229 Slices (12783 LUTs, 326 FFs)	153.846	26
	11	1691 Slices (3365 LUTs, 326 FFs)	208.33	72
[7] DL- PISO GF(2^{163})	41	7385 Slices (13218 LUTs, 378 FFs)	144.927	27.6
	11	1899 Slices (3912 LUTs, 444 FFs)	175.438	85.5
Proposed method GF(2^{163})	**82**	**12075 Slices (23370 LUTs, 490 FFs)**	**207.108**	**14.484**
	41	**6331 Slices (12265 LUTs, 490 FFs)**	**241.061**	**20.74**
	21	**3432 Slices (6640 LUTs, 494 FFs)**	**269.879**	**33.345**
	11	**1960 Slices (3800 LUTs, 502 FFs)**	**316.051**	**53.788**
	6	**1184 Slices (2287 LUTs, 518 FFs)**	**355.075**	**78.856**
Proposed method GF(2^{233})	**117**	**16782 Slices (32866 LUTs, 699 FFs)**	**207.751**	**14.439**
	59	**8536 Slices (16743 LUTs, 699 FFs)**	**262.504**	**19.045**
	30	**4394 Slices (8593 LUTs, 699 FFs)**	**318.210**	**28.287**
	15	**2407 Slices (4657 LUTs, 699 FFs)**	**329.545**	**51.578**
	8	**1458 Slices (2811 LUTs, 699 FFs)**	**394.594**	**83.622**

It should be noted that in [7] number of D flip flops for output register in DL-PISO structure and for serial input (A input) in the DL-SIPO structure have not been considered in the implementation. According to the Table 6, the proposed work has better timing results than [7] on similar FPGA family Virtex-4 XC4VLX100-ff1148. For example in field GF(2^{163}) execution times of the proposed structures are 20.74ns and 53.788ns for two digit sizes 41 and 11, respectively, which are better than execution time in [7] for similar digit sizes. Table 7 shows area, critical path delay, and execution time of the proposed structure in 180nm CMOS technology by Synopsys Design Vision tool. Results show a suitable trade-off between area and execution time, applicable for elliptic curve cryptography systems.

Table 7: ASIC results of the proposed structures

Field	Digit size	Technology	Area (μm^2)	Critical path delay (ns)	Time (ns)
Proposed method GF(2^{163})	**82**	**180nm**	**1364592**	**7.47**	**22.41**
	41	**180nm**	**690407**	**5.15**	**25.75**
	21	**180nm**	**370341**	**3.61**	**32.49**
	11	**180nm**	**212556**	**3.3**	**52.8**
	6	**180nm**	**133525**	**3.13**	**90.77**
Proposed method GF(2^{233})	**117**	**180nm**	**1879851**	**6.78**	**20.34**
	59	**180nm**	**981012**	**5.4**	**27**
	30	**180nm**	**519713**	**4.56**	**41.04**
	15	**180nm**	**284420**	**3.31**	**56.27**
	8	**180nm**	**177167**	**2.47**	**81.51**

4. Conclusions

This paper presents an FPGA and ASIC implementation of an efficient hardware structure of the digit-serial Gaussian normal basis multiplier over GF(2^m). In the proposed structure by reviewing the multiplication equation in normal basis, a regular structure for Gaussian normal basis multiplier is presented. The structure of multiplier is based on exponentiation by powers of 2 and multiplication by normal element of GF(2^m). Therefore, the proposed architecture has low hardware complexity and low critical path delay. It is suitable for high-speed hardware implementation of the finite field multiplication and inversion operations over GF(2^m) for elliptic curve cryptography.

References

[1] D. Hankerson, A. Menezes, S. Vanstone, “Guide to Elliptic Curve Cryptography”, Springer-Verlag, New York, 1st edn., 2004.

[2] Jenn-Shyong Horng, I-Chang Jou and Chiou-Yng Lee, “On complexity of normal basis multiplier using modified Booth’s algorithm”, Proc. of the 7th WSEAS International Conference on Applied Informatics and Communications, Athens, Greece, August 24-26, 2007, pp.12-17.

[3] C.W. Chiou, H.W. Chang, W.-Y.Liang, C.-Y.Lee, J.-M.Lin, Y.-C.Yeh, “Low-complexity Gaussian normal basis multiplieroverGF(2^m)”, IET Inf. Secur., 6(4), 2012, pp. 310-317.

[4] Chiou-Yng Lee,CheWunChiou, “Scalable Gaussian Normal Basis Multipliers over GF(2^m) Using Hankel Matrix-Vector Representation”, J Sign Process Syst (69), 2012, pp. 197-211.

[5] Reza Azarderakhsh and Arash Reyhani-Masoleh, “A Modified Low Complexity Digit-Level Gaussian Normal Basis Multiplier”,Proc. Third Int’l Workshop Arithmetic of Finite Fields (WAIFI), June 2010, pp. 25-40.

[6] A. Reyhani-Masoleh, “Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases”, IEEE Trans. Computers, 55(1), Jan. 2006, pp. 34-47.

[7] R. Azarderakhsh and A. Reyhani-Masoleh, “Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases”, IEEE Trans. Comput., 62(4), Apr. 2013, pp. 744-757.

[8] Jenn-Shyong HORNG, I-Chang JOU, Chiou-Yng LEE, “Low-complexity multiplexer-based normal basis multiplier over GF(2^m)”, J Zhejiang UnivSci A 2009 10(6), pp. 834-842.

[9] T.-P. Chuang, C. WunChiou, S.-S.Lin, C.-Y. Lee, “Fault-tolerant Gaussian normal basis multiplier over GF(2^m)”, IET Inf. Secur., 2012, 6(3), pp. 157-170.

[10] 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.

[11] C¸. K. Koc¸ and B. Sunar, “An Efficient Optimal Normal Basis Type II Multiplier over GF(2^m)” IEEE Trans. Computers, 50(1), Jan. 2001,pp. 83-87.

[12] CheWunChiou, Chiou-Yng Lee and Yun-Chi Yeh, “Sequential Type-I Optimal Normal Basis Multiplier and Multiplicative Inverse in GF(2^m)”, Tamkang Journal of Science and Engineering, 13(4), 2010,pp. 423-432.

[13] A. Reyhani-Masoleh and M.A. Hasan, “Low Complexity Word-Level Sequential Normal Basis Multipliers,” IEEE Trans. Comput., 54(2), Feb. 2005, pp. 98-110.

[14] Zhen Wang, Xiaozhe Wang, and Shuqin Fan, “Concurrent Error Detection Architectures for Field Multiplication Using Gaussian Normal Basis”, Proc. of Information Security, Practice and Experience (ISPEC), LNCS 6047, 2010, pp. 96-109.

[15] CheWunChiou, Jim-Min Lin, Yu-Ku Li, Chiou-Yng Lee, Tai-Pao Chuang, and Yun-Chi Yeh, “Pipeline Design of Bit-Parallel Gaussian Normal Basis Multiplier over GF(2^m)”, Advances in Intelligent Systems and Computing, Springer, 238, 2014, pp. 369-377.

[16] Bayat-Sarmadi, S., Hasan, M.A.: Concurrent Error Detection in Finite-Filed Arithmetic Operations Using Pipelined and Systolic Architectures. IEEE Trans. Comput., 58, 2009, pp. 1553-1567.

[17] Chiou, C. W., Chang, C. C., Lee, C. Y., Lin, J. M., & Hou, T. W., “Concurrent error detection and correction in Gaussian normal basis multiplier over GF(2^m)”, IEEE Trans Comput., 58 (6), 2009, pp. 851-857.

[18] Kwon, S., “A low complexity and a low latency bit parallel systolic multiplier over GF(2^m) using an optimal normal basis of type II”, Proc. of 16th IEEE Symp. Computer Arithmetic, June 2003, pp. 196-202.

[19] 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), 2009, pp. 1-4.

[20] Reza Azarderakhsh, Mehran Mozaffari Kermani, Siavash Bayat-Sarmadi, and Chiou-Yng Lee, “Systolic Gaussian Normal Basis Multiplier Architectures Suitable for High-Performance Applications”, IEEE Trans on Very Large Scale Integration (VLSI) Systems, (99), 2014, pp.1-4.

[21] Yong sukcho, Jae Yeon Choi, “A new Word-parallel bit-serial Normal basis multiplier over GF(2^m)”, International Journal of control and Automation, 6(3), June 2013, pp. 209-216.

[22] Chiou-Yng Lee, “Concurrent error detection architectures for Gaussian normal basis multiplication over GF(2^m)”, Integration, the VLSI journal, 43, 2010, pp. 113-123.

[23] Wang, Z., Fan, S., “Efficient montgomery-based semi-systolic multiplier for even-type GNB of GF(2^m)”, IEEE Trans. Comput., 61(3), 2012, pp. 415-419.

[24] Jeng-Shyang Pan, Chiou-Yng Lee, Yao Li, “Subquadratic space complexity Gaussian normal basis multipliers over GF(2^m) based on Dickson–Karatsuba decomposition”, IET Circuits Devices Syst., 2015, 9(5), pp. 336–342.

[25] D.W. Ash, I.F. Blake, and S.A. Vanstone, “Low Complexity Normal Bases”, Discrete Applied Math., 25, 1989, pp. 191-210.

[26] IEEE P1363: Editorial Contribution to standard for Public Key Cryptography, 2003.

[27] Federal Information Processing Standards Publications (FIPS)186-2, U.S. Department of Commerce/NIST: Digital Signature Standard (DSS), 2000.