High-speed VLSI implementation of Digit-serial Gaussian normal basis Multiplication over GF(2^m)

Bahram Rashidi1, Sayed Masoud Sayedi2, Reza Rezaeian Farahshahi3
1Dept. of Elec. & Comp. Eng., Isfahan University of Technology, Isfahan 84156-83111, Iran
2Dept. of Mathematical Sciences, Isfahan University of Technology, Isfahan 84156-83111, Iran
3The School of Mathematics, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran, Iran
b.rashidi@ec.iut.ac.ir, m_sayedi@cc.iut.ac.ir, farashahi@cc.iut.ac.ir

Abstract—In this paper, by employing the logical effort technique an efficient and high-speed VLSI implementation of the digit-serial Gaussian normal basis multiplier is presented. It is constructed by using AND, XOR and XOR tree components. To have a low-cost implementation with low number of transistors, the block of AND gates are implemented by using NAND gates based on the property of the XOR gates in the XOR tree. To optimally decrease the delay and increase the drive ability of the circuit the logical effort method as an efficient method for sizing the transistors is employed. By using this method and also a 4-input XOR gate structure, the circuit is designed for minimum delay. The digit-serial Gaussian normal basis multiplier is implemented over two binary finite fields GF(2^{65}) and GF(2^{233}) in 0.18μm CMOS technology for three different digit sizes. The results show that the proposed structures, compared to previous structures, have been improved in terms of delay and area parameters.

Keywords: Cryptography, Logical Effort, Gaussian Normal Basis multiplication, digit-serial, VLSI implementation

1. Introduction

Several well-known cryptographic algorithms such as Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), and some message authentication codes are based on the properties of finite fields. In ECC the use of elliptic curves defined over finite fields is proposed for cryptography schemes, including key exchange, encryption, and digital signature. A hierarchy process is used for hardware implementation of ECC. The finite field arithmetic operations such as field multiplication, field squaring, and field inversion are involved in the implementation process. The most important component in this cryptosystem is field multiplication. The binary finite fields, denoted by GF(2^m), are well suited for hardware implementation, where the addition operation is performed by modulo 2 without carry bit. The elements in binary fields are represented by a basis. Two practical bases are polynomial basis (PB) and normal basis (NB). The efficient normal basis for implementation is denoted by Gaussian normal basis (GNB) [1]. The GNB is considered in several standards such as IEEE P1363 [2] and NIST [3]. For example, five NIST recommended fields GF(2^{163}), GF(2^{233}), GF(2^{283}), GF(2^{409}) and GF(2^{571}) are respectively corresponded to the even types T={4,2,6,4, and 10} of GNB. 

In the normal basis representation the powers of two of the field element are implemented only by cyclic shift operation. In the special case, the squaring operation is performed by only one-bit cyclic shift to left. This feature can be useful in the hardware implementation of the field multiplier over normal basis. Many different architectures of the normal basis and GNB multiplications are presented in previous works [4]-[29]. In [6], a novel scalable multiplication algorithm for a type- T Gaussian normal basis by using Hankel Matrix-Vector representation is presented. In [7] a modified digit-level GNB multiplier over GF(2^m) is proposed. Also for types T bigger 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 [9] a low-complexity digit-level serial input parallel output (SIPO) GNB multiplier, also an improved digit-level parallel input serial output (PISO) multiplier architecture, and a hybrid architecture by connecting the output of the digit-level PISO multiplier to the input of the digit-level SIPO multiplier are presented. In [10] a new normal basis multiplication algorithm based on a divide-and-conquer and a uniform shift method is used to implement an efficient multiplier-based architecture. A bit-parallel GNB multiplier based on one pipelined XOR tree is designed in [17]. In [21] a novel algorithm for GNB binary finite field multiplication by using Toeplitz matrix-vector representation is proposed. Multipliers with systolic and semi-systolic architecture are presented in [5], [14], [16], [18], [19], [20] and [22]. The main problem in the systolic structures is their very high hardware consumption and high numbers of clock cycles. In [22] the number of clock cycles is reduced. The focus of this work is on the efficient VLSI implementation of the digit-serial GNB multiplication. Our previously reported digit-serial Gaussian normal basis multiplier [30] is used for implementation. The multiplier has a highly regular structure with low critical path delay and low hardware resources, and it is well suited to hardware implementations. In the multiplier, an XOR tree for summation of partial products exists. A structural VLSI implementation of the XOR tree based on logical effort technique is presented. In the multiplier structure, the generated partial products by AND gates blocks are added to each other by XOR tree, because in the binary field, addition is performed by modulo 2 using XOR gates. Accordingly, the blocks of AND gates are...
implemented by NAND gates based on the property of the XOR tree in the multiplier structure. Also, an
optimized 4-input XOR gate is used for implementation of the XOR tree.

The rest of the paper is organized as follows. Section 2 provides a brief background on Gaussian normal basis
multiplication over GF(2^m) and the structure of digit-serial GNB multiplier. Section 3 describes the proposed
VLSI implementation of the digit-serial GNB multiplier over GF(2^m). Section 4 gives a comparison between this
work and other previously related works. The paper is concluded in section 5.

2. Gaussian normal basis multiplication over GF(2^m) and the structure of digit-serial multiplier

Let β be a normal element of the binary finite field GF(2^m). Then, the set of elements \{β, β^2, ..., β^{2^m-1}\} in
GF(2^m) is a basis for the space GF(2^m) over GF(2). This means each element A ∈ GF(2^m) can be represented by
A = ∑_{i=0}^{m-1} a_i β^{2^i} = (a_0 β^2 + a_1 β^2^2 + a_2 β^{2^3} + ... + a_{m-2} β^{2^{m-2}} + a_{m-1} β^{2^{m-1}}), where a_i are 0 or 1. And, for
the simplicity, the element A is represented by a vector A = [a_{m-1}, a_{m-2}, ..., a_2, a_1, a_0]. The addition of two elements in GF(2^m) is computed by bitwise XOR gates. Also the squaring of the element A is performed as follows:

\[ A^2 = [a_{m-2}, a_{m-3}, ..., a_0, a_{m-1}] \]

One important property of using normal basis representation, as shown in above formula, is performing the
squaring operation very efficiently by a simple one-bit cyclic shift to left. In general case for the positive integer n, the computation of 2^n-th power of the element A is performed via n-bit cyclic shift to left, i.e.,

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

Also 2^{-n}-th power is computed by n-bit cyclic shift to right,

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

The Gaussian normal basis (GNB) is special class of normal basis for low complexity normal basis [1]-[2].
For the binary finite field GF(2^m), where m>1 and is not divisible by 8, and for a positive integer T, let \( p = mT + 1 \)
be a prime number such that \( \gcd(mt, m) = 1 \), where k is the multiplication order of 2 module p. Then there exists
a normal basis over GF(2^m) called the GNB of type T. In GNB the number of nonzero entries of multiplication
matrix is less or equal to \( (mT-1) \). The time and area complexity of the multiplication operation depends on the
type of the normal basis with respect to that basis. In this work, we consider the GNBS with odd values of m
which are applicable for cryptography applications, and it implies that T is an even number.

Here, we briefly discuss the structure of digit-serial Gaussian normal basis multiplier presented in [30]. Let A, B
be two elements in GF(2^m). The element \( B = [b_{m-1}, b_{m-2}, ..., b_2, b_1, b_0] \) is divided into d words of w bits where
d = \( \lceil \frac{m}{w} \rceil \). Then, we have \( \sum_{i=1}^{w} B_i \), where \( B_i = \sum_{k=1}^{d} b_{m-(k-1)w-i} \beta^{2^{m-(k-1)w-i}} \)
for \( i = 1, ..., w \). Here, we set \( b_i = 0 \) if \( i \leq 0 \). The multiplication of elements A, B in GF(2^m) is written by

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

In other words, we have

\[ C = \sum_{i=1}^{w} C_i^{2^{w-i}} = \left( (C_1^2 + C_2^2 + C_3^2 + ...) + C_w \right) \]

where for \( i = 1, ..., w, \)

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

To have a low-complexity and regular architecture of multiplication by \( \beta^{2^{(m-kw)}} \), the computation of \( x \beta^{2^{(m-kw)}} \)
is performed as \( \left( x^{2^{-(m-kw)}} \right) \beta^{2^{(m-kw)}} \) in three steps; first the exponentiation of the input x by \( 2^{-(m-kw)} \)
is done, then multiplication by \( \beta \) is performed, and finally the exponentiation of the result by \( 2^{(m-kw)} \) is
completed. The details of multiplication by \( \beta \) which is the main part of the implementation are given in [30].
In the computation of \( C_i \), the exponentiation by \( 2^{-(m-kw)} \) is performed in the following regular form:
\[(A^{2^{-(w-1)}})^{2^{(l-1)}} \cdot 2^{-(m-kw)} = \ldots \left(\left(A^{2^{-(w-1)}}\right)^{2^{(l-1)}}\right)^{2^{-(m-kw)}} \cdot 2^w\].

In above equation, first exponentiation by \(2^{-(m-kw)}\) is computed, and then for \(k = 2,3,\ldots, d\), exponentiations by \(2^{-(m-kw)}\) are generated by a \(d-1\) length sequence of exponentiation by \(2^w\).

The following example shows the structure of digit-serial GNB multiplier over GF(\(2^7\)) of type \(T=4\) for the case of \(w=3, d = \left\lceil \frac{2}{3} \right\rceil = 3\). In this case, the element \(B\) is represented by three words \(B_1, B_2, B_3\) by

\[B_1 = b_6\beta^{2^6} + b_5\beta^{2^5} + b_4\beta^{2^4},\]
\[B_2 = b_3\beta^{2^3} + b_2\beta^{2^2} + b_1\beta^{2^1},\]
\[B_3 = b_0\beta.\]

The values \(C_i\), for \(i = 1,2,3\), are given as follows.

\[C_1 = \left((A^{2^{-(2)}})^{2^{3}}\right) \cdot \beta b_6 + \left((A^{2^{-(2)}})^{2^{3}}\right)^{2} \cdot \beta b_3 + \left((A^{2^{-(2)}})^{2^{3}}\right)^{2} \cdot \beta b_0\]
\[C_2 = \left((A^{2^{-(2)}})^{2^{4}}\right)^{2} \cdot \beta b_5 + \left((A^{2^{-(2)}})^{2^{4}}\right)^{2} \cdot \beta b_2 + \left((A^{2^{-(2)}})^{2^{4}}\right)^{2} \cdot \beta b_1\]
\[C_3 = \left((A^{2^{-(2)}})^{2^{4}}\right) \cdot \beta b_4 + \left((A^{2^{-(2)}})^{2^{4}}\right)^{2} \cdot \beta b_1 + \left((A^{2^{-(2)}})^{2^{4}}\right)^{2} \cdot \beta b_0\]

The bits \(b_{-1}\) and \(b_{-2}\) are set to zero. The product \(C = AB\) is computed by

\[C = \left(C_1^2 + C_2^2 + C_3\right).\]

Fig. 1 shows the structure of the digit-serial GNB multiplier over GF(\(2^7\)). The required exponentiation operations in the multiplier structure are implemented by wired cyclic shifts.

The critical data path of the structure of digit-serial GNB multiplier over GF(\(2^w\)) with type \(T\) is \(T_A + (|\log_2 T| + |\log_2(T+1)|)T_X\), where \(T_A\) and \(T_X\) denote the time delay of a 2-input AND gate and 2-input XOR gate.
respectively [30]. The digit-serial GNB multiplier requires \( dm \) AND gates and less or equal than \( dm+(T-1)(m-1) \) XOR gates. More details of the hardware and time complexity of this work and related works are presented in [30]. In Fig.5, the critical data path is \( T_A + [\log_2 m] + [\log_2 (3+1)]T_X = T_A + 5T_X \). In [30] the output signal of the multiplier is obtained from flip-flops outputs (see Fig.1 in [30]), so one clock cycle is added to \( d \) clock cycles and the total number of clock cycles is \( d+1 \). In current work, we change the place of D flip-flops, and the output signal of multiplier is obtained from final XOR gates outputs in the XOR tree, as seen in Fig.1. In this case, one clock cycle is reduced and the number of clock cycles is \( d \). This leads to reduction of clock cycles by order \( O(m) \) in computation of point multiplication for the elliptic curve cryptography.

3. Proposed Implementation of Digit-serial GNB Multiplier over \( GF(2^m) \) Based on Logical Effort

The proposed method here is applicable for all multipliers in which an XOR tree is used to perform the sum of the partial products. The examples are bit-serial ONB multipliers with PISO structure [15], [27], bit-parallel and digit-serial ONB and GNB multipliers [7]-[14], [28], [30] and bit-parallel and digit-serial PB multipliers [31]-[34], in the binary finite fields. In these structures to generate partial products, one bit of input operand (for example \( B \)) is ANDed by an \( m \)-bit vector in the structure of multipliers. In this work, in the structure of digit-serial GNB multiplier, the number of AND blocks are equal to \( d \), and each block includes \( m \) 2-input AND gates. One straightforward method for implementation of AND gate is using NAND-NOT structure, in which in CMOS structure, any 2-input AND gate is implemented by 6 transistors (4 transistors for NAND gate and 2 transistors for inverter gate). By considering the structure of the multiplier and properties of the XOR gates, the implementation of the AND gates can be done only by using NAND gates. As seen in the structure of multiplier, the output of AND blocks in the summation part are bit-wise XORed by XOR tree. We use the following equation:

\[
a_k \oplus a_{k-1} \oplus \ldots \oplus a_1 \oplus a_0 = \bar{a}_k \oplus \bar{a}_{k-1} \oplus \ldots \oplus \bar{a}_1 \oplus \bar{a}_0 \quad (1)
\]

which is true if \( k \) is an even number and \( a_i \in \{0, 1\} \), for \( 0 \leq i \leq k \). Based on this equation XORing of even numbers of input bits is equal to XORing of the complemented of same input bits. In the case that \( k \) is an odd number, since \( k-1 \) is even, based on above equation we have:

\[
a_k \oplus a_{k-1} \oplus \ldots \oplus a_1 \oplus a_0 = a_k \oplus \bar{a}_{k-1} \oplus \ldots \oplus \bar{a}_1 \oplus \bar{a}_0 \quad (2)
\]

Fig.2 shows configuration of the AND blocks and XOR tree in the multiplier structure. Output signal of register Reg1 is \( S \) and the \( m \)-bit output of each AND block is called \( AP_i \) where \( 1 \leq i \leq d \).

![Configuration of the AND part and XOR tree in the multiplier structure](image)

For each output bit, there are following cases, if \( d \) is an even number:

\[
\text{Output}(j) = AP_d(j) \oplus AP_{d-1}(j) \ldots AP_1(j) \oplus S(j) = \bar{AP}_d(j) \oplus \bar{AP}_{d-1}(j) \oplus \ldots \oplus \bar{AP}_1(j) \oplus S(j) \quad (3)
\]

and if \( d \) is an odd number we have:

\[
\text{Output}(j) = AP_d(j) \oplus AP_{d-1}(j) \ldots AP_1(j) \oplus S(j) = \bar{AP}_d(j) \oplus \bar{AP}_{d-1}(j) \oplus \ldots \oplus \bar{AP}_1(j) \oplus \bar{S}(j) \quad (4)
\]

where Output\((j)\) is \( j^{th} \) bit of the multiplier output. The schematic of the circuit (for the case of even \( d \)) for one bit of the multiplier output is shown in Fig.3. Original structure of the AND blocks and XOR tree is shown in
Fig. 3 (a). Also Fig. 3 (b) shows the implementation of the AND blocks based on the proposed method by NAND gates. In the implementation of the AND blocks by using only NAND gates the number of inverter gates is reduced by \( m \times d \). The proposed approach for implementation of the AND part is applicable for other GNB and polynomial basis multipliers.

As seen in Fig. 1 for generation of partial products in the structure of multiplier, one bit of input operand, for example \( b_i \), is ANDed by an \( m \)-bit vector. Therefore, one of the input pins of each AND gate must be connected to \( b_i \) input. Fig. 4 (a) shows this concept. Here the \( b_i \) signal must drive \( m \) NAND gates, since we have changed the AND gates by NAND gates. If \( C_{in-NAND} \) is the input capacitance of each NAND gate, then \( b_i \) signal has a capacitance load equal to \( m \times C_{in-NAND} \). In elliptic curve cryptography, the range of \( m \) is bigger than 160, so this capacitance load is very big and it can considerably increase the delay of the circuit. To decrease the delay the \( b_i \) signal is buffered in Fig. 4 (b).

\[ \alpha = \frac{C_{load}}{C_{in1}} \]  

(5)

where \( C_{in1} \) here is the input capacitance of one NAND gate. For minimum delay the number of inverters \( N \) is calculated by:

\[ N = ln \frac{C_{load}}{C_{in1}} \]  

(6)

Equations (5) and (6) are used for the design of the buffer. As an example, the buffer design for field \( GF(2^{33}) \) is presented next. In the proposed structure \( C_{in1}=4.263fF \) and \( C_{load}=m \times C_{in-NAND}=233 \times 3.282fF=765fF \). So we have 6 inverter stages in the structure of the buffer. The value \( \alpha \) is:
The structure of the buffer that drives one NAND block in the digit-serial GNB multiplier over GF(2^{233}) is shown in Fig. 5 (b).

\[
\alpha = \left[ \frac{C_{\text{load}}}{C_{\text{in1}}} \right]^n = \left[ \frac{765.1}{4.263} \right]^6 = 2.38
\]

Fig. 5: (a) structure of super buffer, (b) structure of the buffer for drive of one NAND block over GF(2^{233})

Another part of the multiplier that needs buffering is the output of D flip-flops connected to \( \beta \) blocks. Fig. 6 shows the structure after applying the changes for the implementation over GF(2^n). As seen in the figure, the digit size \( d \) is odd, so based on Eq. (4), for implementation of AND blocks by NAND gates the outputs of D flip-flops (S signals) are inverted.

In Fig. 6 the structure of the digit-serial GNB multiplier is constructed by XOR Tree, \( \beta \) blocks, NAND gates, D flip-flops and multiplexers. The main element of the circuit is XOR gate. A brief discussion on different low-cost full swing circuits of the XOR gates is presented in [36]. In [37] a 6 transistors XOR circuit is designed. In this circuit, for two states of \( A=1' \), \( B=0' \) and \( A=0' \), \( B=1' \) the level of output voltage depends on voltage level of input signals. In [36], a modified version of XOR gate is presented, in which this limitation is eliminated. Fig. 7 (a) shows the modified 2-input XOR structure in which two minimum size pull-up PMOS transistors are added to the previous circuit. In this circuit the XOR output (X) and XNOR output (\( \bar{X} \)) signals are produced simultaneously. This is an important property of the structure for construction of the 4-input XOR gate. Fig. 7 (b) shows the implemented layout of the 2-input XOR gate in 0.18\( \mu \)m CMOS technology.
The 4-input XOR gate is constructed by two modified 2-input XOR gates, pass transistors and one inverter at the output node [36]. Fig.8 (a) shows the circuit, and the layout of the gate is shown in Fig.8 (b). As seen in the figure two output signals \( X \) and \( \overline{X} \) of the 2-input XOR gates are used for construction of the 4-input XOR gate. Voltage level and driving capability is restored by using an inverter at the output node. The effect of process variations and mismatch on the delay of the circuit was evaluated through the Monte Carlo analysis. Fig.9 shows the result for 500 iterations and for load capacitance of 30fF. As the figure shows the mean value of the delay is 225.191ps.

Another component used in the multipliers structure is D flip-flop. Different static flip-flop topologies have been proposed in the past. Based on a comparative study, some of the widely used topologies have been shown in Fig.10 (a)-(c). A full characterization of these flip-flops can be found in the literature [38]-[41]. Here, a brief description of each flip-flop is presented. Fig.10 (a) shows a conventional Transmission Gate Flip-Flop (TGFF). This structure requires a large number of transistors for implementation. A Push Pull Flip-Flop (PPFF) is shown in Fig.10 (b) in which an inverter and a transmission gate between the outputs of the master and slave latches
accomplish a push pull effect at the slave latch. Fig.10 (c) shows a Clocked CMOS (C²MOS) flip-flop. In the figure, the second and fourth C²MOS latches are used to maintain the charge levels at output nodes. These latches are weak feedback latches with low driving capability. There are 20 transistors used in this circuit, which is high compared to that of two other circuits.

The Transmission Gate Master-Slave flip-flop (TG-MSFF) [42] shown in Fig.11 (a) is used in current work for implementation of digit-serial GNB multiplier. In the circuit when the clock signal is high ‘1’, the first transmission gate in the master part becomes functional samples and transfers input data at node D to the inverter output node. When the clock goes ‘0’, the second transmission gate in the slave part becomes functional transfers the data from intermediate inverter output node to the output node Q. This structure is one of the fastest classical structures [43]. It has a short direct path (low latency direct path) and a low power feedback, which are constructed by cascading two identical pass gate latches. Based on simulation results presented in [39], [43], [44], [45], the best power-performance trade-off with total delay (clock-to-output + setup time) is achieved for this structure. As mentioned before, in this paper, we use this D flip-flop in the proposed implementation of the digit-serial GNB multiplier. Fig.11 (b) shows the implemented layout of the TG-MSFF in 0.18μm CMOS technology.

Based on above structures, implementations of the multiplier over two practical fields GF(2^{163}) and GF(2^{233}) for three digit sizes $d=3$, $d=15$ and $d=59$ are done. In the following, implementations of the GNB multiplier over field GF(2^{233}) based on logical effort technique is presented. The logical effort technique, describes capability of one logic gate relative to that of a reference inverter gate [46]. The logical effort of a logic gate is defined as the ratio of its input capacitance to that of an inverter that delivers equal output current. The logical effort parameters are presented briefly in [36]. Here, the logical effort is applied to get least overall delay by balancing the delay among the stages. First implementation of multiplier for $(d=59, w=4)$ is described. In the general case as shown in Fig.12 the XOR tree is implemented in six stages by using 2-input XOR gates. The low-cost 4-input XOR gate with minimum number of transistors is used to implement the XOR tree. The 6-stage XOR tree in the digit-serial GNB for case $(d=59, w=4)$ over GF(2^{233}) is implemented by three levels of 4-input XOR gates. Each 4-input XOR gate constructed by two logic stages including an inverter and a 2-input XOR gate and pass transistors. The sizes of transistors are computed for different electrical effort by using logical effort technique. The process for the 6-stage structure of the XOR tree in the digit-serial GNB multiplier $(d=59, w=4)$ over GF(2^{233}) which is shown in Fig.13 for $H=10$ is described in details in the following.

Fig.10: Three different structures of D flip-flop presented in literature.
For the 6-stage structure, the path logical effort is the product of logical efforts of three inverters and three XOR gates, calculated as \( G = g_1 g_2 \cdots g_6 = \left( \frac{1.4252}{2.22} \right)^3 \times (1.1682)^3 = 1.5942 \). The branching effort is \( B = 1 \), because there is no branching along the path; so the path effort is \( F = GBH = 1.5942 \times 10 = 15.942 \). Minimum delay can be realized if the transistors sizes in each stage are chosen properly. To that end first the stage effort is computed as \( \hat{f} = \sqrt[6]{10 \times 1.1682^7} = 1.588 \). Since \( H \) is equal to 10, \( C_{in} = 10C_0 = 14.252 \text{fF} \) where \( C_0 \) is the input capacitance of the 4-input XOR gate. The input capacitance of each gate is computed by Eq. (1) in [36]. It can be started with the load capacitance at the output node of the path. The method is a backward approach as follows:

\[
\begin{align*}
C_{in,6} &= C_{out,6} \times \frac{g_6}{f} = 14.252 \text{fF} \times \left( \frac{1}{1.588} \right) = 8.98 \text{fF} \Rightarrow W_{n,12} = 1.62 \mu\text{m}, W_{p,12} = 3.24 \mu\text{m}, \\
C_{in,5} &= C_{in,6} \times \frac{g_5}{f} = 8.98 \text{fF} \times \left( \frac{1.1682}{1.588} \right) = 6.61 \text{fF} \Rightarrow W_{n,4} = W_{p,5} = W_{p,5} = W_{p,5} = 1 \mu\text{m}, \\
C_{in,4} &= C_{in,5} \times \frac{g_4}{f} = 6.61 \text{fF} \times \left( \frac{1}{1.588} \right) = 4.164 \text{fF} \Rightarrow W_{n,4} = 0.75 \mu\text{m}, W_{p,4} = 1.5 \mu\text{m},
\end{align*}
\]
As it was expected, the size of the computed first stage input capacitance is equal to the input capacitance of 4-input XOR gate at first stage. \(W_{n1}, W_{n2}, W_{p1},\) and \(W_{p2}\) are the sizes of input nMOS and pMOS transistors in the 4-input XOR gate. Also \(W_{p1}\) and \(W_{p2}\) are the sizes of nMOS and pMOS transistors of the inverter in the 4-input XOR. Based on above calculations transistors of output stages are wider, which enable them to drive current into large output loads.

For the case \((d=15, w=16)\) over \(GF(2^{233})\) as seen in Fig.14 (a), the proposed XOR tree is implemented by two levels 4-input XOR gate in 4 logic stages. Fig.14 (b) shows implementation of the circuit by using 2-input XOR gates in a general 4 logic stages. The path logical effort of the proposed 4-stage XOR tree structure for this case is \(G=g_1g_2...g_8=(1.4252)^2\cdot(1.1682)^7=1.365\), and for the electrical effort of \(H=50\) the value of path effort is computed as \(F=GBH=68.25\). The sizes of transistors for minimum delay are calculated as follows. The stage effort is \(f = \sqrt{10\times1.1682^2} = 2.874\). Starting with the output load \(50C_c=71.26fF\), Eq. (1) in [36] is applied to compute input capacitances of the stages as follows:

\[
C_{in-4} = C_{out-4} \cdot \frac{g_8}{f} = 71.26fF \times \frac{1}{2.874} = 24.8fF \Rightarrow W_{n4} = 4.5\mu m, \ W_{p4} = 9\mu m,
\]
\[
C_{in-3} = C_{in-4} \cdot \frac{g_7}{f} = 24.8fF \times \frac{1.682}{2.874} = 10.08fF \Rightarrow W_{n3} = W_{n2} = W_{p3} = W_{p2} = 1.57\mu m,
\]
\[
C_{in-2} = C_{in-3} \cdot \frac{g_6}{f} = 10.08fF \times \frac{1}{2.874} = 1.931fF \Rightarrow W_{n2} = 0.64\mu m, \ W_{p2} = 1.28\mu m,
\]
\[
C_{in-1} = C_{in-2} \cdot \frac{g_5}{f} = 3.51fF \times \frac{1.682}{2.874} = 1.426fF \Rightarrow W_{n1} = W_{n2} = W_{p1} = W_{p2} = 0.22\mu m.
\]

To evaluate the performance of the circuit, the layout of the digit-serial GNB multiplier \((d=3, w=78)\) over \(GF(2^{233})\) with 2-stage XOR tree was implemented and post-layout simulation applied. Fig.15 shows the structure of multiplier for this case.
Fig. 14: (a) implementation of the XOR tree for digit-serial GNB multiplier \( (d=15, w=16) \) over \( \text{GF}(2^{233}) \) by the 4-input XOR gate in 4 logic stages, (b) by using 2-input XOR gates in general 5 logic stages.

In the layout design, proper distribution of the clock signal is an important subject. Here, the main aim in clock distribution is transmitting the clock signal simultaneously to all D flip-flops. There are different clock distribution methods such as tree buffer, mesh, H-tree and some combinations of them [47]-[49] that are used to achieve zero clock skewing. In H-tree clock distribution approach, which is a common zero skew routing method, by matching the length of each path, from clock source to D flip-flop, zero skew clock routing is achieved. This is performed by creating a series of routes with “H” shape. At the corners of each “H” the nearly identical clock signals, provide the inputs to the next level of smaller “H” routes. To minimize reflections, the impedance of interconnects are scaled. For an H-tree network, each route leaving a junction must have twice the impedance of the source route. This is accomplished by decreasing the interconnect width of each route. This continues until the final points of the H-tree structure are used to drive either the D flip-flops, or local buffers that drive the D flip-flops. In this work, we consider H-tree distribution for clock signal. Fig.16 shows the topology of clock distribution network based on H-tree method for the proposed structure of the digit-serial GNB multiplier \( (d=3, w=78) \) over \( \text{GF}(2^{233}) \).

Fig.15: Structure of the digit-serial GNB multiplier \( (d=3, w=78) \) over \( \text{GF}(2^{233}) \) for layout implementation
The layout of the proposed structure of the digit-serial GNB multiplier ($d=3$, $w=78$) over $GF(2^{233})$ for the case of $H=10$, $C_L=14.252fF$ is shown in Fig.17. The area of the layout is $790\mu m \times 798\mu m$. Result of Monte Carlo analysis for the delay of the circuit is shown in Fig.18. The number of iterations is $N=300$. As the figure shows the mean value of delay is 580.048ps.
The proposed structures were successfully implemented in 0.18μm CMOS technology. The parameters of proposed structures and a comparison between present work and other implementations of the multiplier over GF(2^m) are presented in Tables 1 and 2. The comparisons are based on parameters of critical path delay, calculation time and area. The implementations are presented for different electrical efforts, namely, H=10, 50 and 250. Table 1 shows the results for the proposed implementation of digit-serial GNB multiplier over GF(2^{10}) and GF(2^{233}) for both cases of applying the logical effort technique and without applying it. Table 2 compares time and area of the proposed structures and some previously reported structures. The reported simulation results are based on schematic structures. The areas are estimated by summation of transistors area without considering the routing and the buffers in clock distribution. Only for the case of 2-stage structure of the digit-serial GNB multiplier (d=3, w=78) for H=10 and C_L=14.252fF the results are obtained from post layout simulation.

**Table 1: Critical path delay, time and area results for the proposed implementation of digit-serial GNB multiplier**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Field</th>
<th>C_L/H</th>
<th>Critical Path Delay (ns)</th>
<th>Time (ns)</th>
<th>Area (μm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{10})</td>
<td>14.252fF/10</td>
<td>3.268</td>
<td>9.204</td>
<td>2085043</td>
</tr>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>4.415</td>
<td>12.645</td>
<td>2085043</td>
</tr>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{10})</td>
<td>356.3fF/250</td>
<td>6.797</td>
<td>19.791</td>
<td>2085043</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{10})</td>
<td>14.252fF/10</td>
<td>1.843</td>
<td>20.273</td>
<td>364070</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>2.916</td>
<td>28.787</td>
<td>564070</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>5.520</td>
<td>36.52</td>
<td>564070</td>
</tr>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>3.011</td>
<td>11.244</td>
<td>1771633</td>
</tr>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>4.238</td>
<td>16.952</td>
<td>1771633</td>
</tr>
<tr>
<td>6-stage, d=59</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>6.638</td>
<td>26.148</td>
<td>1771633</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>1.510</td>
<td>24.16</td>
<td>498980</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>2.584</td>
<td>38.144</td>
<td>498980</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>3.347</td>
<td>50.352</td>
<td>498980</td>
</tr>
<tr>
<td>5-stage, d=15</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>0.850</td>
<td>66.3</td>
<td>163668</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>1.428</td>
<td>4.284</td>
<td>2060919</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>1.623</td>
<td>4.869</td>
<td>2064179</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{10})</td>
<td>356.3fF/250</td>
<td>2.043</td>
<td>6.129</td>
<td>2088488</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>1.356</td>
<td>14.916</td>
<td>550867</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>1.785</td>
<td>19.635</td>
<td>552497</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>2.231</td>
<td>24.541</td>
<td>571236</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>1.262</td>
<td>5.048</td>
<td>1737149</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>1.484</td>
<td>5.936</td>
<td>1741809</td>
</tr>
<tr>
<td>Proposed 6-stage with LE, d=59</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>2.013</td>
<td>8.052</td>
<td>1775128</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>1.018</td>
<td>16.288</td>
<td>480107</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>71.26fF/50</td>
<td>1.451</td>
<td>23.216</td>
<td>482437</td>
</tr>
<tr>
<td>Proposed 4-stage with LE, d=15</td>
<td>GF(2^{233})</td>
<td>356.3fF/250</td>
<td>1.880</td>
<td>30.08</td>
<td>503786</td>
</tr>
<tr>
<td>Proposed 2-stage with LE, d=3</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>0.375</td>
<td>29.25</td>
<td>159772</td>
</tr>
<tr>
<td>Proposed 2-stage with LE, d=3</td>
<td>GF(2^{233})</td>
<td>14.252fF/10</td>
<td>0.563</td>
<td>43.914</td>
<td>630420</td>
</tr>
</tbody>
</table>

In this case, results are achieved from post-layout simulation.
In this work when applying...

Concurrent Error Detection Architectures for...

As the results show...

\[ \begin{align*} 
\text{Table 2: Comparison of time and area of the proposed structure and other implementations of the multiplier} \\
\begin{array}{|c|c|c|c|c|} 
\hline
\text{Methods} & \text{Field} & \text{Technology} & \text{C/H} & \text{Time (ns)} & \text{Area (\mu m^2)} \\
\hline
[9] d=11, GNB, DL-SIPO & GF(2^{163}) & 65nm & --- & 13.95 & 34278 \\
[9] d=11, GNB, DL-PISO & GF(2^{233}) & 65nm & --- & 20.70 & 34837 \\
[9] d=55, GNB, DL-PISO & GF(2^{233}) & 65nm & --- & 10.65 & 161495 \\
\hline 
\text{Proposed 6-stage with LE, d=59} & GF(2^{163}) & 180mm & 14.252fF/10 & 4.284 & 2060919 \\
\text{Proposed 4-stage with LE, d=15} & GF(2^{233}) & 180mm & 14.252fF/10 & 14.916 & 550867 \\
\text{Proposed 6-stage with LE, d=59} & GF(2^{233}) & 180mm & 14.252fF/10 & 5.048 & 1737149 \\
\text{Proposed 4-stage with LE, d=15} & GF(2^{233}) & 180mm & 14.252fF/10 & 16.288 & 460107 \\
\text{Proposed 2-stage with LE, d=3} & GF(2^{233}) & 180mm & 14.252fF/10 & 29.25 & 159772 \\
\text{Proposed 2-stage with LE, d=3 \downarrow} & GF(2^{233}) & 180mm & 14.252fF/10 & 43.914 & 630420 \\
\hline
\end{array}
\end{align*} 
\]

In this case, results are achieved form post-layout simulation.

5. Conclusions

An efficient VLSI implementation of the digit-serial Gaussian normal basis multipliers was presented. The proposed methods are general and applicable for high-speed hardware implementation of the multiplication operation over binary finite fields. In the proposed structures by using, logical effort technique, 4-input XOR gate, and implementation of the AND gate blocks by using NAND gates, speed and area of the multiplier over GF(2^{163}) and GF(2^{233}) have been improved. The proposed implementation is applicable for ASIC implementation of the elliptic curves cryptosystems.

References


