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

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
[3]The School of Mathematics, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran, 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, 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($2^{163}$) 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 multiplexer-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 $\beta$ be a normal element of the binary finite field GF($2^m$). Then, the set of elements $\{\beta, \beta^2, \dots, \beta^{2^{m-1}}\}$ in GF($2^m$) is a basis for the space GF($2^m$) over GF(2). This means each element $A \in$GF($2^m$) can be represented 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 $a_i$ are 0 or 1. And, for the simplicity, the element $A$ is represented by a vector $A = [a_{m-1}, a_{m-2}, \dots, a_2, a_1, a_0]$. 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:

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

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.,

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

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

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

The 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($\frac{mT}{k}$,$m$) =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 $B = [b_{m-1}, b_{m-2}, \dots, b_2, b_1, b_0]$ is divided into $d$ words of $w$ bits where $d = \left\lceil \frac{m}{w} \right\rceil$. 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 $i = 1, \dots, w$. Here, we set $b_i = 0$ if $i \leq 0$. The multiplication of elements $A, B$ in GF($2^m$) is written by

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

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 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 $\left(\left(x^{2^{-(m-kw)}}\right)\beta\right)^{2^{(m-kw)}}$ 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:

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

In above equation, first exponentiation by $2^{-(m-w)}$ is computed, and then for $k = 2,3,\dots,d$, 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, $d = \left\lceil\frac{7}{3}\right\rceil$=3. In this case, the element $B$ is represented by three words $B_1, B_1, B_3$ 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.

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

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

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

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

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

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.

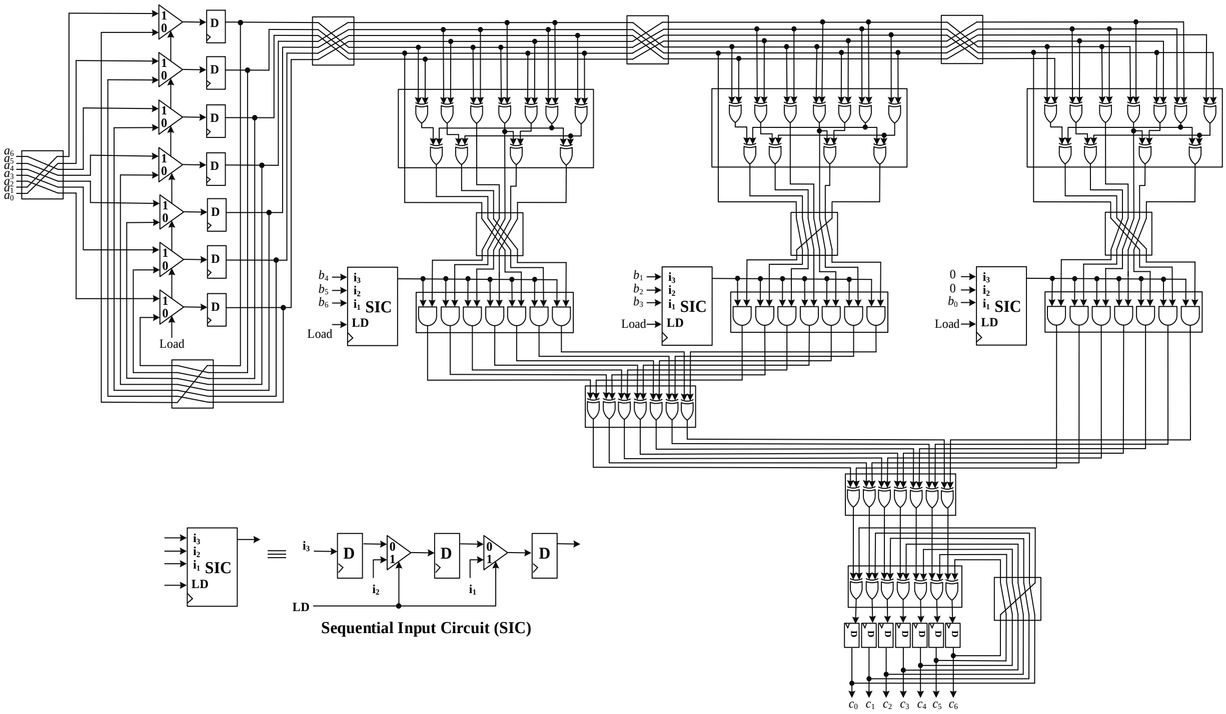


Fig.1: Structure of the digit-serial GNB multiplier over GF($2^7$) with $w$=3 and $d$=3

The critical data path of the 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 [30]. The digit-serial GNB multiplier requires $dm$ AND gates and less or equal than $dm$+($T$-1)($m$-1)$d$ XOR gates. More details of the hardware and time complexity of this work and other related works are presented in [30]. In Fig.5, the critical data path is $\mathrm{T_A} + (\lceil log_2^4 \rceil + \lceil log_2^{(3+1)} \rceil)\mathrm{T_X} = \mathrm{T_A} + 5\mathrm{T_X}$. 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 $0 \le i \le k$. 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 $\mathrm{AP}_i$ where $1 \le i \le d$.

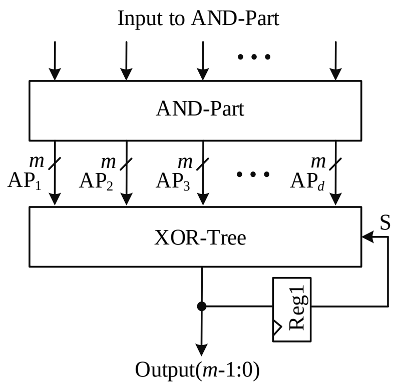

Fig.2: Configuration of the AND part and XOR tree in the multiplier structure

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

$$\mathrm{Output}(j) = \mathrm{AP}_d(j) \oplus \mathrm{AP}_{d-1}(j) \ldots \mathrm{AP}_1(j) \oplus \mathrm{S}(j) = \overline{\mathrm{AP}}_d(j) \oplus \overline{\mathrm{AP}}_{d-1}(j) \oplus \ldots \oplus \overline{\mathrm{AP}}_1(j) \oplus \mathrm{S}(j) \quad (3)$$

and if $d$ is an odd number we have:

$$\mathrm{Output}(j) = \mathrm{AP}_d(j) \oplus \mathrm{AP}_{d-1}(j) \ldots \mathrm{AP}_1(j) \oplus \mathrm{S}(j) = \overline{\mathrm{AP}}_d(j) \oplus \overline{\mathrm{AP}}_{d-1}(j) \oplus \ldots \oplus \overline{\mathrm{AP}}_1(j) \oplus \overline{\mathrm{S}}(j) \quad (4)$$

where $\mathrm{Output}(j)$ 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.

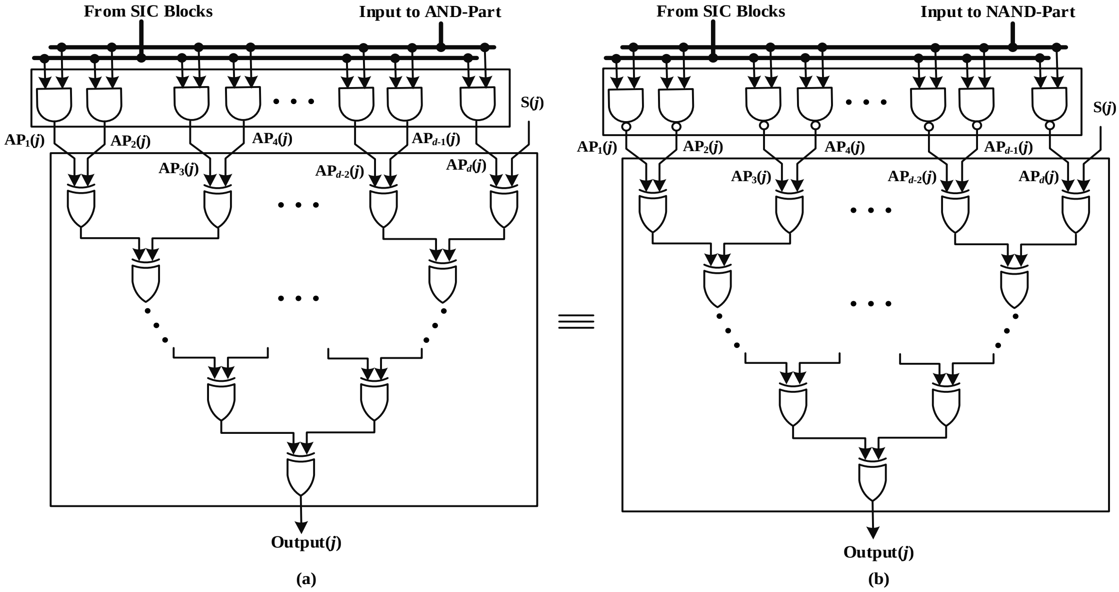


Fig.3: (a) original structure of the AND blocks and XOR tree, (b) and implementation of the AND blocks based on the proposed method by NAND gates, for $j^{th}$ bit of the multiplier output.

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 $C_{\text{in-NAND}}$ is the input capacitance of each NAND gate, then $b_i$ signal has a capacitance load equal to $m{\times}C_{\text{in-NAND}}$. 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).

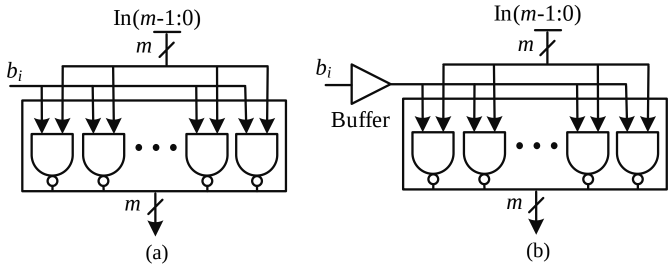


Fig.4: (a) NAND block, (b) NAND block when $b_i$ is buffered

Supper buffers are used to drive large capacitance loads for minimum delay. In the following, we discuss the design of a buffer based on a cascade of N inverters to drive a capacitance load, $C_{\text{load}}$ [35]. Each inverter in the chain is larger than the previous one by a factor $\boldsymbol{\alpha}$. Fig.5 (a) shows this structure. The factor $\boldsymbol{\alpha}$ is calculated as follows:

$$\boldsymbol{\alpha}=\left[\frac{C_{\text{load}}}{C_{in1}}\right]^{\frac{1}{\mathbf{N}}} \tag{5}$$

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

$$\mathbf{N}=ln\frac{C_{\text{load}}}{C_{in1}} \tag{6}$$

Equations (5) and (6) are used for the design of the buffer. As an example, the buffer design for field GF($2^{233}$) is presented next . In the proposed structure $C_{in1}$=4.263fF and $C_{\text{load}}=m{\times}C_{\text{in-NAND}}$=233×(3.282fF)=765fF.

$\mathbf{N}=ln\frac{C_{\text{load}}}{C_{in1}}= ln\frac{765}{4.263} = 5.2{\sim}6$. So we have 6 inverter stages in the structure of the buffer. The value $\boldsymbol{\alpha}$ is:

$$\alpha=\left[\frac{C_{load}}{C_{in1}}\right]^{\frac{1}{N}}=\left[\frac{765}{4.263}\right]^{\frac{1}{6}}=2.38$$

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).

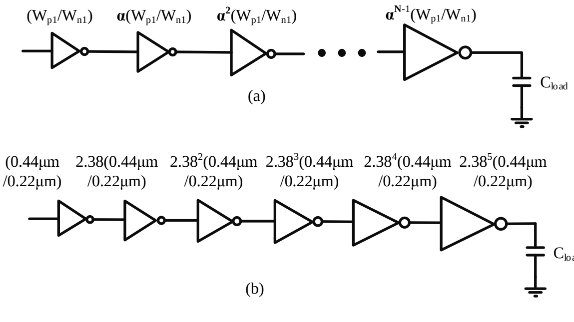


Fig.5: (a) structure of supper buffer, (b) structure of the buffer for drive of one NAND block over GF($2^{233}$)

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($2^7$). 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.

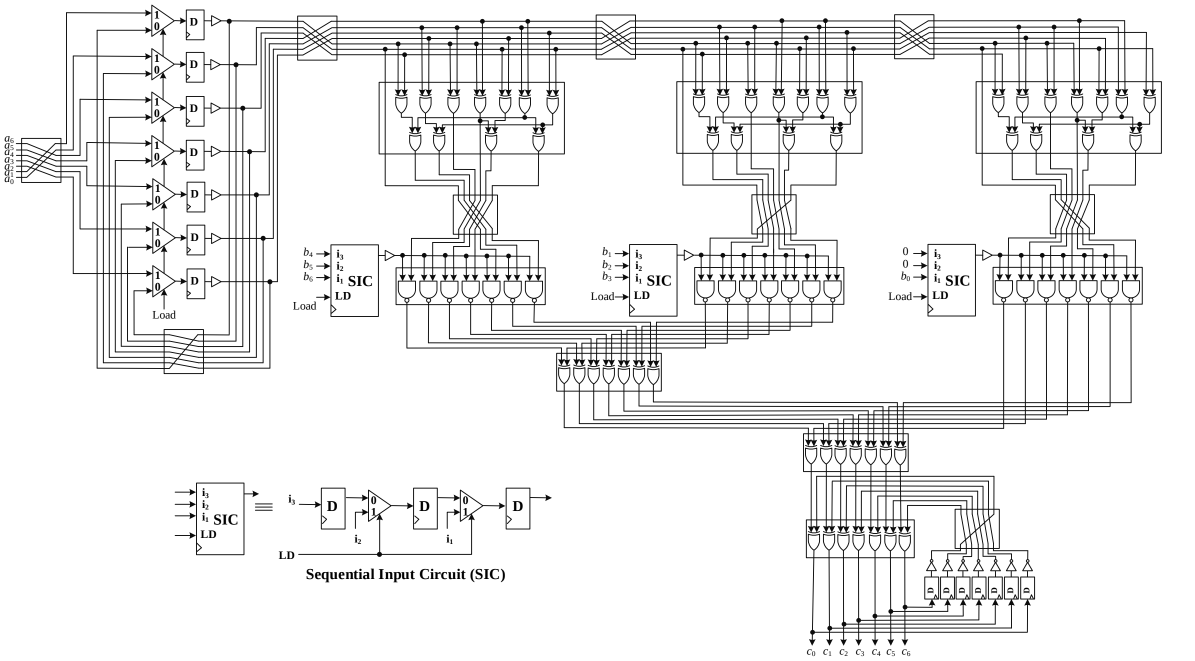


Fig.6: Proposed structure for GF($2^7$) after applying changes for VLSI implementation

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 ($\overline{X}$) 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 0.18µm CMOS technology.

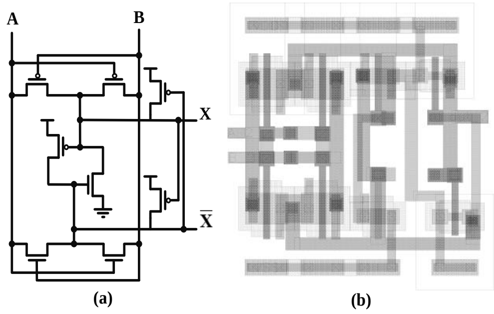


Fig.7: (a) 2-input XOR structure in [36], (b) layout of this gate

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.

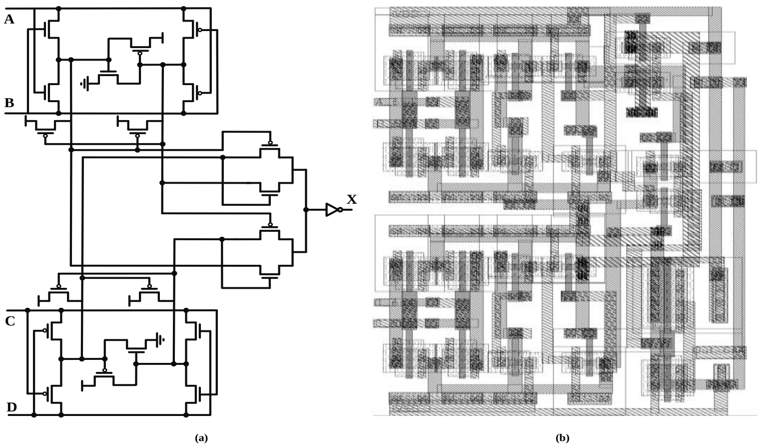


Fig.8: (a) the structure of the 4-input XOR gate in [36], (b) layout of this gate

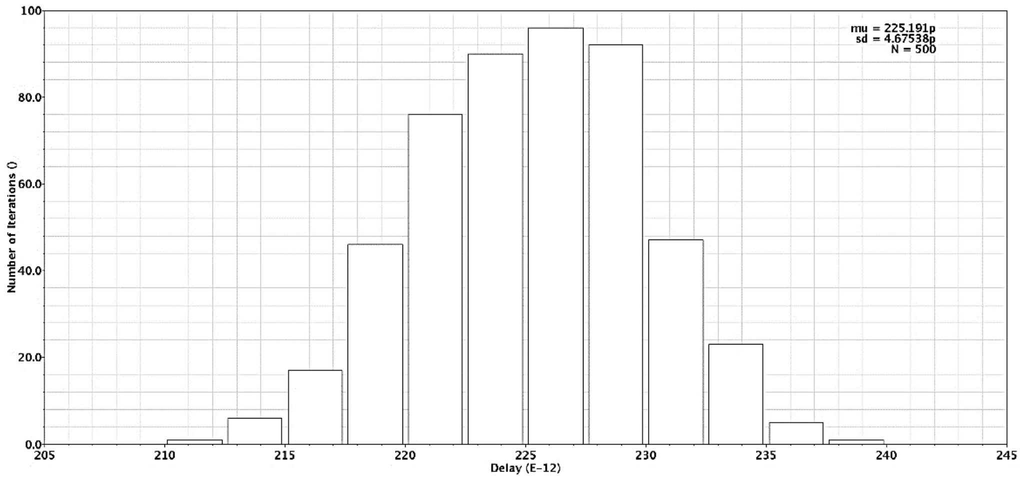


Fig.9: Monte Carlo result for the delay of the 4-input XOR circuit

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 ($C^2MOS$) flip-flop. In the figure, the second and fourth $C^2MOS$ 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.

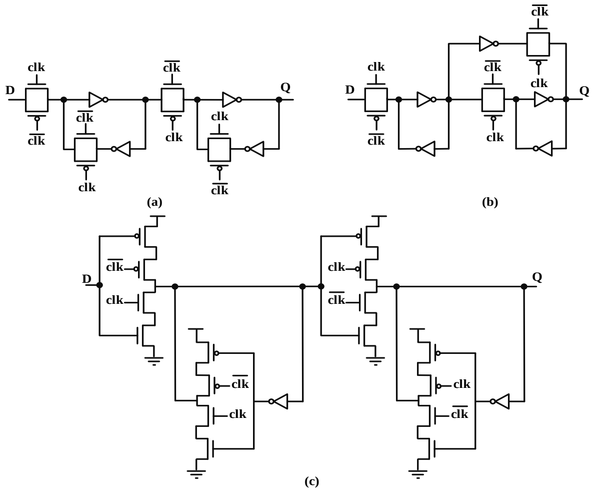


Fig.10: Three different structures of D flip-flop presented in literature.

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.

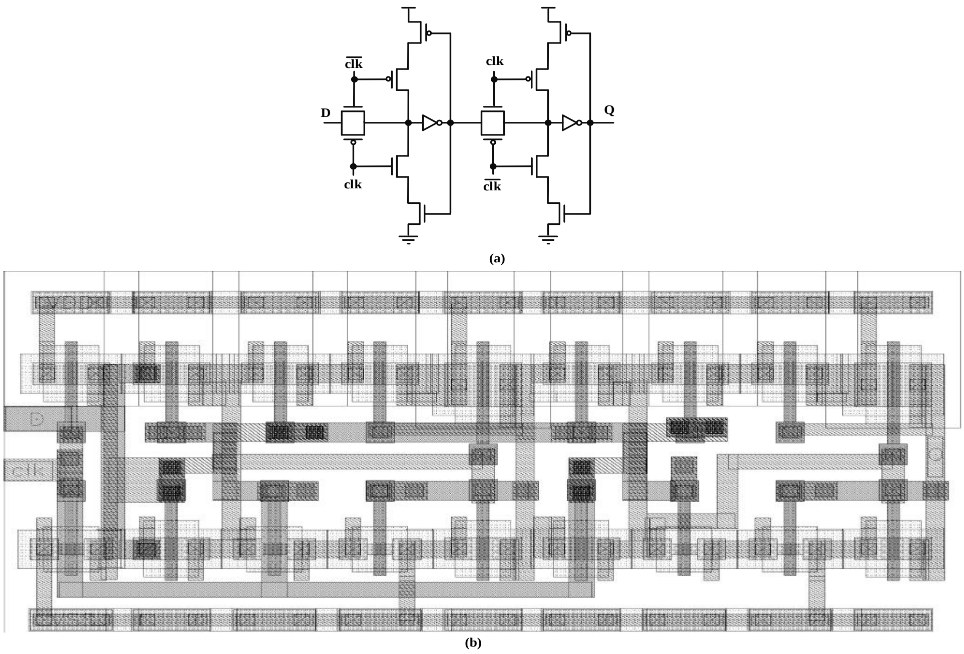


Fig.11: (a) the structure of the Transmission Gate Master-Slave D flip-flop circuit, (b) the implemented layout of TG-MSFF

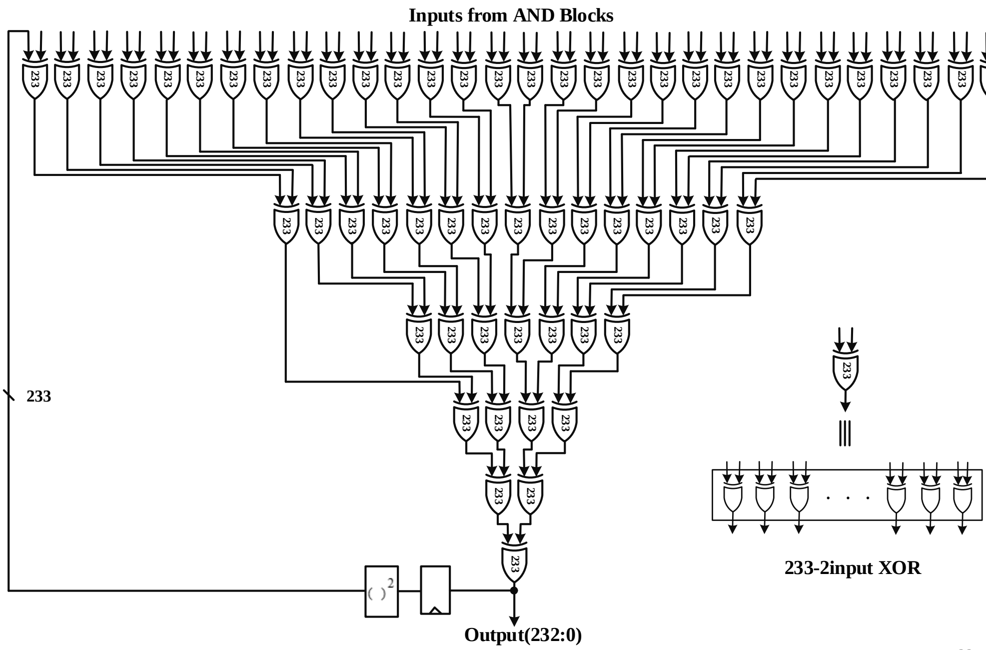


Fig.12: General 6-stage structure of the XOR tree in the digit-serial GNB multiplier ($d$=59, $w$=4) over GF($2^{233}$) based on 2-input XOR gates

For the 6-stage structure, the path logical effort is the product of logical efforts of three inverters and three XOR gates, calculated as $G=g_1g_2\dots g_6=(\frac{1.4252}{1.22})^3\times(1)^3=(1.1682)^3=1.5942$. The branching effort is B=1, because there is no branching along the path; so the path effort is $F=GBH=1.5942\times10=15.942$. 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 $\hat{f}=\sqrt[6]{10\times1.1682^3}=1.588$. Since H is equal to 10, $C_L=10C_{in}=10C_x=14.252$fF where $C_x$ 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:

$C_{in\text{-}6} = C_{out\text{-}6}\times\frac{g_6}{\hat{f}}=14.252\text{fF}\times(\frac{1}{1.588})=8.98\text{fF} \Rightarrow W_{n\text{-}12}=1.62\mu\text{m}, W_{p\text{-}12} = 3.24\mu\text{m}$,

$C_{in\text{-}5} = C_{in\text{-}6}\times\frac{g_5}{\hat{f}}=8.98\text{fF}\times(\frac{1.1682}{1.588}) = 6.61\text{fF} \Rightarrow W_{n1\text{-}5} = W_{n2\text{-}5} = W_{p1\text{-}5} = W_{p2\text{-}5} = 1\mu\text{m}$,

$C_{in\text{-}4} = C_{in\text{-}5}\times\frac{g_4}{\hat{f}}=6.61\text{fF}\times(\frac{1}{1.588}) = 4.164\text{fF} \Rightarrow W_{n\text{-}4} = 0.75\mu\text{m}, W_{p\text{-}4} = 1.5\mu\text{m}$,

$C_{in-3} = C_{in-4}\times\frac{g_3}{\hat{f}}$=4.164fF×($\frac{1.1682}{1.588}$) = 3.065fF => $W_{n1-3} = W_{n2-3} = W_{p1-3} = W_{p2-3}$ =0.47μm,
$C_{in-2} = C_{in-3}\times\frac{g_2}{\hat{f}}$=3.065fF×($\frac{1}{1.588}$) = 1.931fF => $W_{n-2}$ = 0.35μm, $W_{p-2}$ = 0.7μm,
$C_{in-1} = C_{in-2}\times\frac{g_1}{\hat{f}}$=1.931fF×($\frac{1.1682}{1.588}$) = 1.4212fF = $C_{in}$=> $W_{n1-1} = W_{n2-1} = W_{p1-1} = W_{p2-1}$ =0.22μm.

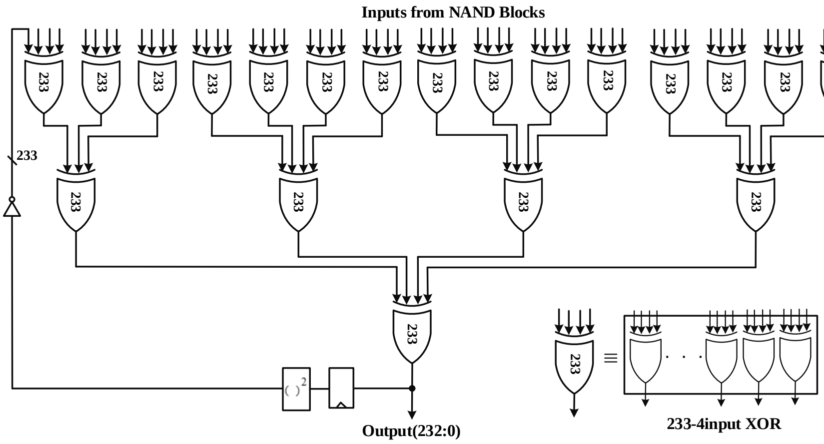

Fig.13: Proposed 6-stage structure of the XOR tree for digit-serial GNB multiplier ($d$=59, $w$=4) over GF($2^{233}$) based on 4-input XOR gate

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. $W_{n1-i}$, $W_{n2-i}$, $W_{p1-i}$ and $W_{p1-i}$ are the sizes of input nMOS and pMOS transistors in the 4-input XOR gate. Also $W_{n-i}$ and $W_{p-i}$ 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 $G=g_1g_2\dots g_6=(\frac{1.4252}{1.22})^2\times(1)^2=(1.1682)^2=1.365$, 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 $\hat{f} = \sqrt[4]{10\times1.1682^2}$ =2.874. Starting with the output load 50$C_x$=71.26fF, Eq. (1) in [36] is applied to compute input capacitances of the stages as follows:

$C_{in-4} = C_{out-4}\times\frac{g_4}{\hat{f}}$=71.26fF× ($\frac{1}{2.874}$) = 24.8fF => $W_{n-4}$ = 4.5μm, $W_{p-4}$ = 9μm,
$C_{in-3} = C_{in-4}\times\frac{g_3}{\hat{f}}$=24.8fF× ($\frac{1.1682}{2.874}$) = 10.08fF => $W_{n1-3} = W_{n2-3} = W_{p1-3} = W_{p2-3}$ =1.57μm,
$C_{in-2} = C_{in-3}\times\frac{g_2}{\hat{f}}$=10.08fF× ($\frac{1}{2.874}$) = 1.931fF => $W_{n-2}$ = 0.64μm, $W_{p-2}$ = 1.28μm,
$C_{in-1} = C_{in-2}\times\frac{g_1}{\hat{f}}$=3.51fF× ($\frac{1.1682}{2.874}$) = 1.426fF = $C_{in}$=> $W_{n1-1} = W_{n2-1} = W_{p1-1} = W_{p2-1}$ =0.22μm.

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.

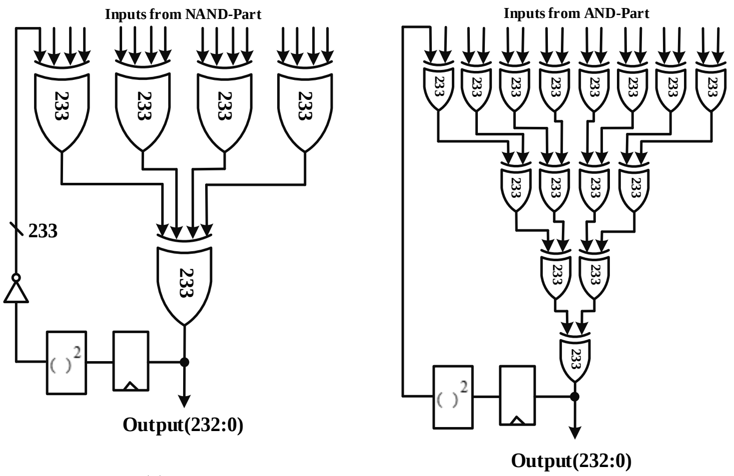


Fig.14: (a) implementation of the XOR tree for digit-serial GNB multiplier ($d$=15, $w$=16) over GF($2^{233}$) by the 4-input XOR gate in 4 logic stages, (b) by using 2-input XOR gates in general 5 logic stages.

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 GF($2^{233}$).

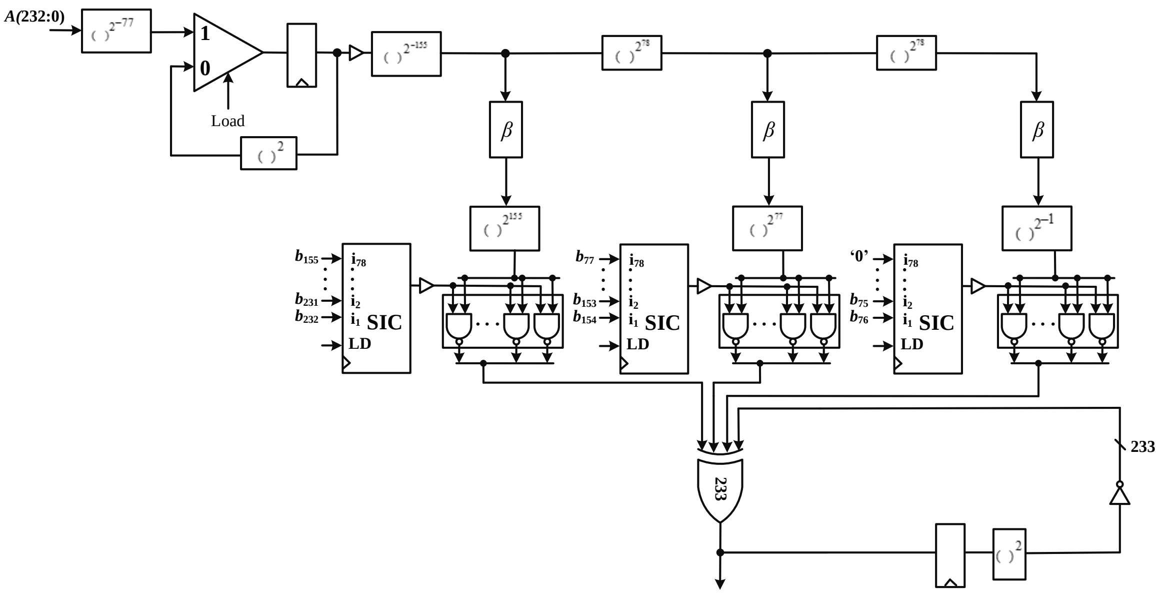


Fig.15: Structure of the digit-serial GNB multiplier ($d$=3, $w$=78) over GF($2^{233}$) for layout implementation

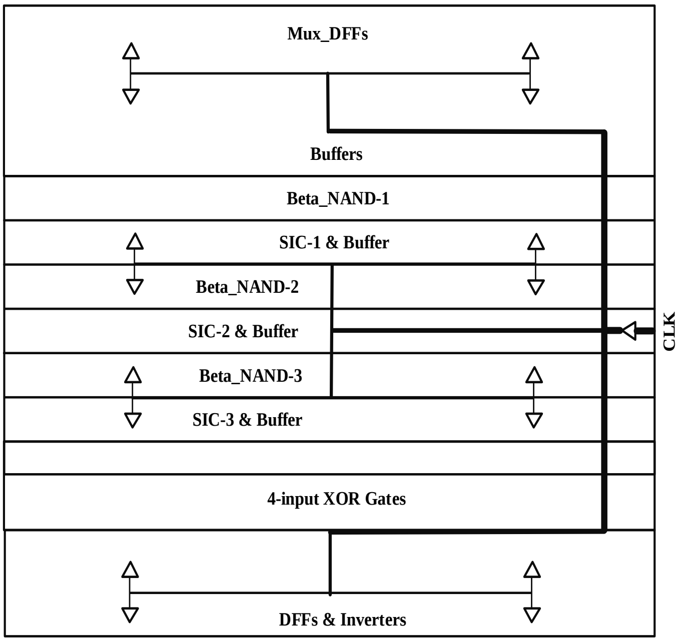

Fig.16: Topology of clock distribution network based on H-tree in the structure of the digit-serial GNB multiplier ($d$=3, $w$=78) over GF($2^{233}$)

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 790µm*798µm. 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.

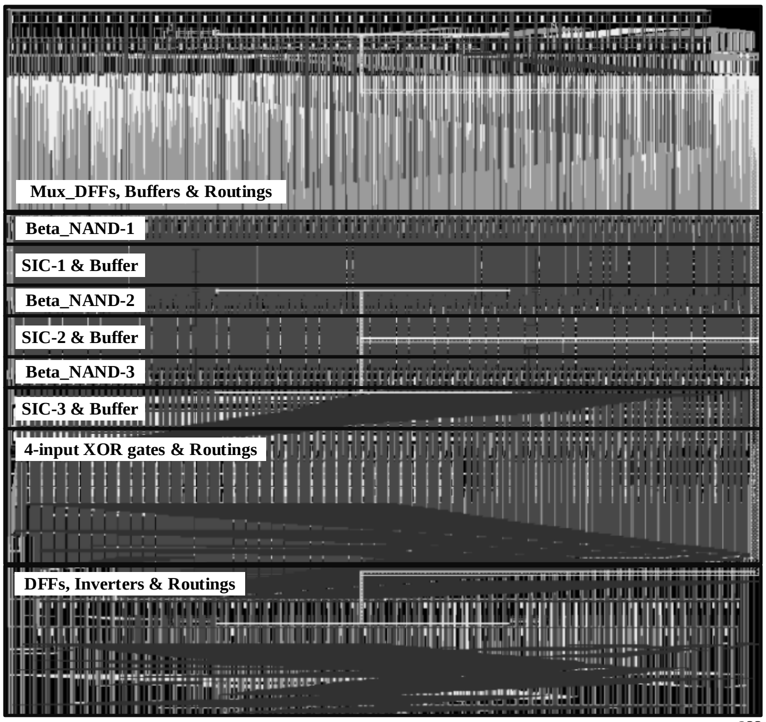

Fig.17: Layout of the proposed structure of the digit-serial GNB multiplier ($d$=3, $w$=78) over GF($2^{233}$) for the case of H=10, $C_L$=14.252fF.

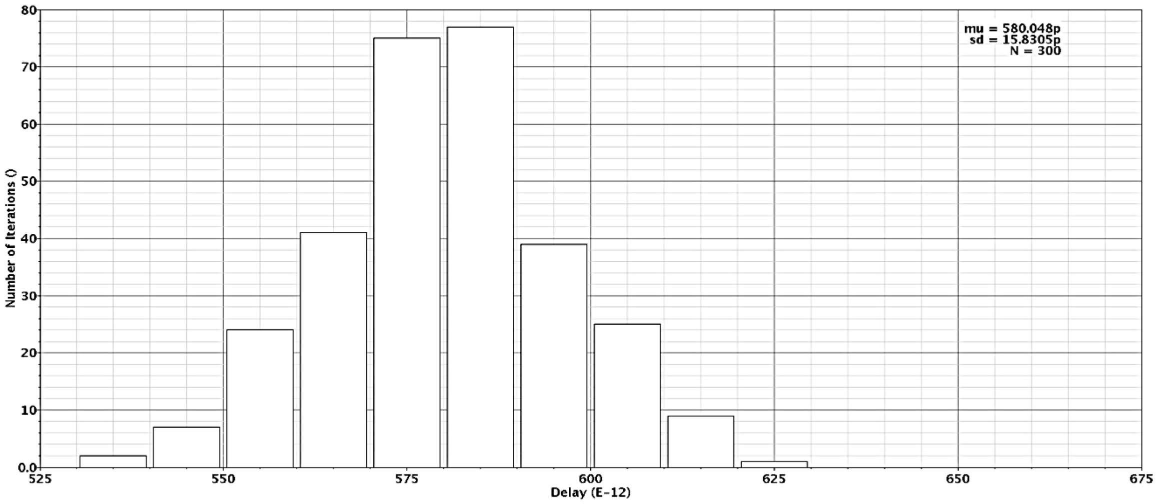


Fig.18: Result of Monte Carlo analysis for the delay of the proposed digit-serial GNB multiplier ($d$=3, $w$=78) over GF($2^{233}$) for the case of H=10, $C_L$=14.252fF and for 300 iterations.

## 4. Results and Comparison

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 Tables1 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($2^{163}$) 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

| Methods | Field | $C_L$/ H | Critical Path Delay (ns) | Time (ns) | Area(μm²) |
|---|---|---|---|---|---|
| 6-stage, $d$=59 | GF($2^{163}$) | 14.252fF/10 | 3.268 | 9.204 | 2085043 |
| 6-stage, $d$=59 | GF($2^{163}$) | 71.26fF/50 | 4.415 | 12.645 | 2085043 |
| 6-stage, $d$=59 | GF($2^{163}$) | 356.3fF/250 | 6.797 | 19.791 | 2085043 |
| 5-stage, $d$=15 | GF($2^{163}$) | 14.252fF/10 | 1.843 | 20.273 | 564070 |
| 5-stage, $d$=15 | GF($2^{163}$) | 71.26fF/50 | 2.916 | 28.787 | 564070 |
| 5-stage, $d$=15 | GF($2^{163}$) | 356.3fF/250 | 3.520 | 36.52 | 564070 |
| 6-stage, $d$=59 | GF($2^{233}$) | 14.252fF/10 | 3.011 | 11.244 | 1771633 |
| 6-stage, $d$=59 | GF($2^{233}$) | 71.26fF/50 | 4.238 | 16.952 | 1771633 |
| 6-stage, $d$=59 | GF($2^{233}$) | 356.3fF/250 | 6.638 | 26.148 | 1771633 |
| 5-stage, $d$=15 | GF($2^{233}$) | 14.252fF/10 | 1.510 | 24.16 | 498980 |
| 5-stage, $d$=15 | GF($2^{233}$) | 71.26fF/50 | 2.584 | 38.144 | 498980 |
| 5-stage, $d$=15 | GF($2^{233}$) | 356.3fF/250 | 3.347 | 50.352 | 498980 |
| 2-stage, $d$=3 | GF($2^{233}$) | 14.252fF/10 | 0.850 | 66.3 | 160368 |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{163}$)** | **14.252fF/10** | **1.428** | **4.284** | **2060919** |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{163}$)** | **71.26fF/50** | **1.623** | **4.869** | **2064179** |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{163}$)** | **356.3fF/250** | **2.043** | **6.129** | **2088488** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{163}$)** | **14.252fF/10** | **1.356** | **14.916** | **550867** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{163}$)** | **71.26fF/50** | **1.785** | **19.635** | **552497** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{163}$)** | **356.3fF/250** | **2.231** | **24.541** | **571236** |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{233}$)** | **14.252fF/10** | **1.262** | **5.048** | **1737149** |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{233}$)** | **71.26fF/50** | **1.484** | **5.936** | **1741809** |
| **Proposed 6-stage with LE, $d$=59** | **GF($2^{233}$)** | **356.3fF/250** | **2.013** | **8.052** | **1775128** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{233}$)** | **14.252fF/10** | **1.018** | **16.288** | **480107** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{233}$)** | **71.26fF/50** | **1.451** | **23.216** | **482437** |
| **Proposed 4-stage with LE, $d$=15** | **GF($2^{233}$)** | **356.3fF/250** | **1.880** | **30.08** | **503786** |
| **Proposed 2-stage with LE,$d$=3** | **GF($2^{233}$)** | **14.252fF/10** | **0.375** | **29.25** | **159772** |
| **Proposed 2-stage with LE,$d$=3 [1]** | **GF($2^{233}$)** | **14.252fF/10** | **0.563** | **43.914** | **630420** |

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

Table 2: Comparison of time and area of the proposed structure and other implementations of the multiplier

| Methods | Field | Technology | $C_L$/ H | Time (ns) | Area($\mu m^2$) |
|---|---|---|---|---|---|
| [9] *d*=11, GNB, DL-SIPO | $GF(2^{163})$ | 65nm | --- | 13.95 | 34278 |
| [9] *d*=55, GNB, DL-SIPO | $GF(2^{163})$ | 65nm | --- | 9.75 | 160298 |
| [9] *d*=11, GNB, DL-PISO | $GF(2^{163})$ | 65nm | --- | 20.70 | 34837 |
| [9] *d*=55, GNB, DL-PISO | $GF(2^{163})$ | 65nm | --- | 10.65 | 161495 |
| [23] GNB, *d*=28 | $GF(2^{163})$ | 65nm | --- | 15.2 | 39091 |
| **Proposed 6-stage with LE, *d*=59** | **$GF(2^{163})$** | 180nm | **14.252fF/10** | **4.284** | **2060919** |
| **Proposed 4-stage with LE, *d*=15** | **$GF(2^{163})$** | 180nm | **14.252fF/10** | **14.916** | **550867** |
| **Proposed 6-stage with LE, *d*=59** | **$GF(2^{233})$** | 180nm | **14.252fF/10** | **5.048** | **1737149** |
| **Proposed 4-stage with LE, *d*=15** | **$GF(2^{233})$** | 180nm | **14.252fF/10** | **16.288** | **480107** |
| **Proposed 2-stage with LE, *d*=3** | **$GF(2^{233})$** | 180nm | **14.252fF/10** | **29.25** | **159772** |
| **Proposed 2-stage with LE, *d*=3 [1]** | **$GF(2^{233})$** | 180nm | **14.252fF/10** | **43.914** | **630420** |

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

In [9] and [23] results of time and area are obtained by automatic synthesis tool Design Vision without layout. As the results show the proposed structures presented in this work when applying the logical effort technique and the 4-input XOR gate have better results compared to the general structures.

## 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


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

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

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

[4] Horng, J.S., Jou, I.C. and Lee, C.Y., “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.

[5] Chiou, C.W., Chang, H.W., Liang, W.Y., Lee,C.Y., Lin, J.M., Yeh, Y.C., “Low-complexity Gaussian normal basis multiplier over $GF(2^m)$”, IET Inf. Secur., Vol. 6, No. 4, 2012, pp. 310-317.

[6] Lee C.Y., Chiou, C.W., “Scalable Gaussian Normal Basis Multipliers over $GF(2^m)$ Using Hankel Matrix-Vector Representation”, J Sign Process Syst, Vol. 69, No. 2, 2012, pp. 197-211.

[7] Azarderakhsh, R., and Reyhani-Masoleh, A., “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.

[8] Reyhani-Masoleh, A., “Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases”, IEEE Trans. Computers, Vol. 55, No. 1, Jan. 2006, pp. 34-47.

[9] Azarderakhsh, R., and Reyhani-Masoleh, A., “Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases”, IEEE Trans. Comput., Vol. 62, No. 4, Apr. 2013, pp. 744-757.

[10] Koc C.K. and Sunar, B., “An Efficient Optimal Normal Basis Type II Multiplier over $GF(2^m)$”, IEEE Trans. Computers, Vol. 50, No. 1, Jan. 2001, pp. 83-87.

[11] WunChiou, C., Lin, J.M., Li, Y.K., Lee, C.Y., Chuang, T.P., and Yeh, Y.C., “Pipeline Design of Bit-Parallel Gaussian Normal Basis Multiplier over $GF(2^m)$”, Advances in Intelligent Systems and Computing, Springer, Vol. 238, 2014, pp. 369-377.

[12] Sukcho, Y., Yeon Choi, J., “A new Word-parallel bit-serial Normal basis multiplier over $GF(2^m)$”, International Journal of control and Automation, Vol. 6, No. 3, June 2013, pp. 209-216.

[13] Horng, J.S., Jou, I.C. and Lee, C.Y., “Low-complexity multiplexer-based normal basis multiplier over $GF(2^m)$”, J Zhejiang Univ Sci, Vol. 10, No.6, 2009, pp. 834-842.

[14] Chuang, T.P., Wun Chiou, C., Lin, S.S., Lee, C.Y., “Fault-tolerant Gaussian normal basis multiplier over $GF(2^m)$”, IET Inf. Secur., 2012, Vol. 6, No. 3, pp. 157-170.

[15] Reyhani-Masoleh, A., and Hasan, M.A., “Efficient Digit-serial Normal Basis Multipliers over Binary Extension Fields”, ACM Trans. Embedded Computing Systems, Vol. 3, No. 3, Aug. 2004, pp. 575-592.

[16] Wun Chiou, C., Lee, C.Y., and Yeh, Y.C., “Sequential Type-I Optimal Normal Basis Multiplier and Multiplicative Inverse in $GF(2^m)$”, Tamkang Journal of Science and Engineering, Vol. 13, No. 4, 2010,pp. 423-432.

[17] Reyhani-Masoleh, A. and Hasan, M.A., “Low Complexity Word-Level Sequential Normal Basis Multipliers”, IEEE Trans. Comput., Vol. 54, No. 2, Feb. 2005, pp. 98-110.

[18] Wang, Z., Wang, X., and Fan, S., “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.

[19] Bayat-Sarmadi, S., Hasan, M.A, “Concurrent Error Detection in Finite-Filed Arithmetic Operations Using Pipelined and Systolic Architectures”, IEEE Trans. Comput., Vol. 58, No. 11, 2009, pp. 1553-1567.

[20] Chiou, C.W., Chang, C.C., Lee, C.Y., Lin, J.M., and Hou, T.W., “Concurrent error detection and correction in Gaussian normal basis multiplier over $GF(2^m)$”, IEEE Trans. Comput., Vol. 58, No. 6, 2009, pp. 851-857.

[21] 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.

[22] Lee, C. and Chang, P., “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.

[23] Azarderakhsh, R., Mozaffari Kermani, M., Bayat-Sarmadi, S., and Lee, C.Y., “Systolic Gaussian Normal Basis Multiplier Architectures Suitable for High-Performance Applications”, IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 23, No. 9, 2014, pp.1969-1972.

[24] Lee, C.Y., “Concurrent error detection architectures for Gaussian normal basis multiplication over GF($2^m$)”, Integration, the VLSI journal, Vol. 43, No. 1, 2010, pp. 113-123.

[25] Wang, Z., Fan, S., “Efficient Montgomery-based semi-systolic multiplier for even-type GNB of GF($2^m$)”, IEEE Trans. Comput., Vol. 61, No. 3, 2012, pp. 415-419.

[26] Hua Li, Chang Nian Zhang, “Low-Complexity Versatile Finite Field Multiplier in Normal Basis”, EURASIP Journal on Applied Signal Processing 9, 2002, pp. 954-960.

[27] A. Reyhani-Masoleh and M. A. Hasan, “A new construction of Massey-Omura parallel multiplier over GF($2^m$)” IEEE Trans. Computers, Vol. 51, 2002, pp. 511-520.

[28] Reyhani-Masoleh, A. and Hasan, M. A., “Low Complexity Word-Level Sequential Normal Basis Multipliers,” IEEE Trans. Computers, Vol. 54, 2005, pp. 98-110.

[29] W. Tang, H. Wu, and M. Ahmadi, “VLSI implementation of bit-parallel word-serial multiplier in GF ($2^{233}$)”, Proceedings of the 3rd International EEE-NEWCAS Conference, June 2005, pp. 399-402.

[30] Bahram Rashidi, Sayed Masoud Sayedi, Reza Rezaeian Farashahi, “Efficient and Low-complexity Hardware Architecture of Gaussian Normal Basis Multiplication over GF($2^m$) for Elliptic Curve Cryptosystems”, IET Circuits Devices Syst., Vol. 10, Iss. 4, 2016, pp. 1-10.

[31] Huapeng Wu, “Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis”, IEEE Transactions on Computers, Vol. 51, No. 7, July 2002, pp. 750-758.

[32] Arash Reyhani-Masoleh, and M. Anwar Hasan, “Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF($2^m$)”, IEEE Transactions on Computers, Vol. 53, No. 8, August 2004, pp. 945-959.

[33] Bahram Rashidi, Sayed Masoud Sayedi, Reza Rezaeian Farashahi, “Efficient implementation of bit-parallel fault tolerant polynomial basis multiplication and squaring over GF($2^m$)”, IET Comput. Digit. Tech., 2015, pp. 1-12.

[34] Bahram Rashidi, Reza Rezaeian Farashahi, Sayed Masoud Sayedi, “Efficient Implementation of Low Time Complexity and Pipelined Bit-Parallel Polynomial Basis Multiplier over Binary Finite Fields”, the ISC Int'l Journal of Information Security, Vol.7, No.2, 2015, pp. 101-114.

[35] R. Jacob Baker, “CMOS Circuit Design, Layout, and Simulation”, IEEE Press Series on Microelectronic Systems, John Wiley & Sons, Inc., Hoboken, New Jersey, 3st edn, 2010.

[36] Bahram Rashidi, Sayed Masoud Sayedi, Reza Rezaeian Farashahi, “An efficient and high-speed VLSI implementation of optimal normal basis multiplication over GF($2^m$) ”, Integration, the VLSI Journal, Vol. 55, 2016, pp. 138-154.

[37] D. Radhakrishanan, “Low-voltage low-power CMOS full adder”, in Proc. of IEE Circuits Devices System, Vol. 148, 2001, pp. 19-24.

[38] Saeeid TahmasbiOskuii, “Comparative study on low-power high-performance flip-flops”, Master Thesis, Linköping University, 2003.

[39] Massimo Alioto, Elio Consoli, Gaetano Palumbo, “Flip-Flop Design in Nanometer CMOS”, Springer International Publishing Switzerland 2015.

[40] U. Ko and P.T. Balsara,“High-Performance Energy-Efficient D-Flip-Flop Circuits”, IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 1, 2000, pp. 94-98.

[41] Imran Ahmed Khan, Mirza Tariq Beg, “A New Area and Power Efficient Single Edge Triggered Flip-Flop Structure for Low Data Activity and High Frequency Applications”, Innovative Systems Design and Engineering, Vol.4, No.1, 2013, pp. 1-12.

[42] G. Gerosa, S. Gary, C. Dietz, D. Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, T. Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kathle, “2.2 W, 80 MHz superscalar RISC processor”, IEEE J. Solid-State Circuits, Vol. 29, Dec. 1994, pp. 1440-1454.

[43] V. Stojanovic and V.G. Oklobdzija, “Comparative Analysis of Master-Slave Latches and Flip-Flops for High-Performance and Low-Power Systems”, IEEE Journal of Solid-State Circuits, Vol. 34, No. 4, 1999, pp. 536-548.

[44] K. Singh, S.C. Tiwari and M. Gupta, “A Master-Slave Flip Flop for Low Voltage Systems with Improved Power-Delay Product”, World Applied Sciences Journal, Vol. 16, 2012, pp. 45-52.

[45] Veladimir Stojanovic and Vojion G. oklobdzija, “Comparative analysis of Master-Slave latches and flip-flops for high-performance and low-power system”, IEEE Journal of Solid-State circuits, Vol. 34, 1999, pp. 536-548.

[46] Ivan Sutherland and R.F. Sproull, “Logical effort: Designing for speed on the back of an Envelope”, IEEE Advanced Research in VLSI, MIT Press, March 1991.

[47] Hucydides Xanthopoulos, “Clocking in Modern VLSI Systems”, Series on Integrated Circuits and Systems, Springer, 2009.

[48] Eby G. Friedman, “Clock Distribution Networks in Synchronous Digital Integrated Circuits”, Proceedings of the IEEE, Vol. 89, No. 5, May 2001, pp. 665-692.

[49] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young, “Clock Generation and Distribution for the First IA-64 Microprocessor” IEEE J. Solid-State Circuits, Vol. 35, 2000, pp. 1545-1552.