Efficient VLSI Implementation of a Sequential Finite Field Multiplier Using Reordered Normal Basis in Domino Logic

Parham Hosseinzadeh Namin, Crystal Roma, Roberto Muscedere, and Majid Ahmadi, *Fellow, IEEE*

***Abstract*— In this paper, a high-speed power-efficient VLSI implementation of a finite field multiplier in GF(2^m) is presented. The proposed design has a serial-in parallel-out architecture and performs the multiplication operation using a reordered normal basis. The basic idea is to implement the main building block of the multiplier in domino logic to reduce the critical path delay. Reduction in dynamic power consumption is achieved by limiting the contention current between the keeper transistor and the pull-down network at the beginning of the evaluation phase by employing a new keeper control circuit. The semi-custom layout of the multiplier was realized in 65-nm CMOS technology. The post place-and-route simulations showed that the multiplier can perform multiplication correctly up to a clock rate of 3.85 GHz and consumes marginally less power than the static CMOS counterpart (also implemented with custom placement and route). The size of the multiplier is currently recommended by the National Institute of Standards and Technology for binary field multiplication in elliptic curve cryptography. The proposed design methodology can also be used in the implementation of similar finite field multipliers possessing regular architectures.**

***Index Terms*— Domino logic, elliptic curve cryptography (ECC), finite field arithmetic, reordered normal basis (RNB), serial-in parallel-out (SIPO) finite field multiplier.**

I. Introduction

EFFICIENT computations of finite field arithmetic are highly important in cryptographic applications where field operations are extensively used, namely, elliptic curve cryptography (ECC) and Elgamal cryptosystem [1]. The binary extension field GF(2^m) is a closed set of 2^m elements, meaning that arithmetic operations over the field elements are conducted without leaving the set [2]. Each element of a finite field can be expressed by a bit sequence of length m. A field can be thought of as a vector space spanned by a vector set of m linearly independent elements, called a basis. Choosing the basis by which field elements are represented plays an important role in the efficient implementation of finite field operations. A number of bases over finite fields have been proposed in the literature, among which polynomial basis (PB) and normal basis (NB) are primarily used in practice [1]. Although the use of PB is most common in software implementations, NB offers a virtually cost-free squaring operation performed by a single cyclic shift over the field element's coordinates, thus making it the better choice for hardware implementation. Among the set of finite field arithmetic operations, the efficient implementation of field multiplication is of upmost importance, as field operations of greater complexity (e.g., exponentiation and division) can be performed by the consecutive use of field multiplication.

It is proven that an NB exists for every field in GF(2^m) [3]. In general, the multiplication operation in NB can be modeled as a matrix-vector multiplication, where a matrix multiplication is required to be performed for each of the product coordinates [3]. The hardware complexity of the multiplication operation is directly affected by the number of nonzero elements inside the multiplication matrix. This number is referred to as the *complexity of NB* and is denoted by C_N. For a given m, C_N varies between the two extreme values of $2m+1$ and m^2 and is minimal in the case of two subclasses of NB, known as type I and II optimal NBs (ONBs) [3]. Gao and Vanstone [4] were the first to present the mathematical formulation for reordered NB (RNB) for the subclass of NB in which a type-II ONB exists [4]. RNB can effectively simplify the multiplication operation by defining it as a closed-form formula rather than a matrix operation.

A fully parallel architecture would be a natural choice for applications in which speed is of great priority. Additionally, by cryptographic standards, the use of high-order fields ($m > 160$) is recommended to ensure a high level of security [1]. However, considering the fact that a parallel architecture has an area complexity of $O(m^2)$, a large m will result in a big, power greedy design not suitable for resource-constrained applications. Contrastingly, a fully serial (sequential) multiplier has an area complexity of $O(m)$, resulting in a significantly smaller structure. Despite their smaller size, sequential multipliers require m clock cycles to complete a full multiplication operation as compared to only one cycle in the case of a fully parallel architecture. Thus, it is desirable to reduce the multiplication delay of a sequential multiplier to compensate for this shortcoming.

Manuscript received November 28, 2017; revised March 20, 2018 and June 6, 2018; accepted June 8, 2018. This work was supported by the Natural Sciences and Engineering Council of Canada. *(Corresponding author: Parham Hosseinzadeh Namin.)*

P. Hosseinzadeh Namin was with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada. He is now with Advanced Macro Devices, Thornhill, ON L3T 7X6, Canada (e-mail: hosseinp@uwindsor.ca).

C. Roma was with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada. She is now with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

R. Muscedere and M. Ahmadi are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: rmusced@uwindsor.ca; ahmadi@uwindsor.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2851958

TABLE I
GATE-LEVEL COMPLEXITY COMPARISON BETWEEN DIFFERENT BIT-SERIAL RNB/TYPE-II ONB MULTIPLIERS OVER $\mathbb{F}_{2^m}$

Multiplier	**# AND**	**# XOR**	**# Reg**	**Critical Path Delay (T_{cp})**	**Clock Cycles**
MO [5]	$2m-1$	$2m-2$	$2m$	$T_A+(\lceil\log_2(2m-1)\rceil)T_X$	m
IMO [6]	m	$2m-2$	$2m$	$T_A+(1+\lceil\log_2 m)\rceil)T_X$	m
Agnew [7]	m	$2m-1$	$3m$	T_A+2T_X	m
Feng [8]	$2m-1$	$3m-2$	$3m-2$	T_A+4T_X	m
b-SMPO I [9]	$\lceil\frac{m}{2}\rceil+1$	$3m$	$3m$	T_A+3T_X	m
b-SMPO II [9]	m	$\lceil\frac{m}{2}\rceil+m$	$3m$	T_A+3T_X	m
XEDS [10]	$2m-1$	$2m-2$	$2m$	$T_A+(\lceil\log_2 2m-1)\rceil)T_X$	m
AEDS [10]	m	$3m-3$	$2m$	$T_A+(\lceil\log_2 2m-1)\rceil)T_X$	m
DLGM$_S$ [11]	m	$2m-2$	$2m$	$T_A+(1+\lceil\log_2 m)\rceil)T_X$	m
DLGM$_P$ [11]	m	$\frac{1}{2}(3m-1)$	$3m$	T_A+2T_X	m
Azarderakhsh [12]	m	$2m-1$	$2m$	T_A+2T_X	m
Kwon [13]	m	$\frac{1}{2}(3m-1)$	$3m$	T_A+2T_X	m
Yang [14]	m	$\frac{1}{2}(3m-1)$	$3m$	T_A+2T_X	m
SIPO [15]	m	$2m$	$3m+1$	T_A+2T_X	m
PISO [15]	m	$2m-1$	$2m+1$	$T_A+(1+\lceil\log_2 m\rceil)T_X$	m

In this paper, we present an optimized VLSI implementation of a serial-in parallel-out (SIPO) RNB multiplier in GF(2^m). Our design is based on the sequential architecture proposed by Wu *et al.* [15]. Originating from an inherent feature of RNB, this architecture has a highly regular structure. The regularity of this architecture has been previously exploited to construct a high-speed custom-layout multiplier by implementing the main building block of the architecture in domino logic [16]. However, this performance improvement in terms of critical path delay is obtained at the cost of a significant increase in power consumption. This is the major drawback characteristic to domino logic circuits. The main objective of this paper is to further improve the performance of the multiplier by employing a custom-designed domino logic circuit that effectively reduces the power dissipation of the domino circuit. It is shown that the new implementation significantly increases the maximum operating frequency compared to its equivalent static CMOS realization, as well as successfully reduces the power consumption to a comparable level.

The organization of this paper is as follows. Section II provides a brief review of RNB and multiplication operations using this basis. A short complexity comparison between existing RNB/type-II ONB multipliers is given in Section III. In Section IV, a new domino logic design for the main building block of the multiplier is presented. VLSI implementation and performance comparisons are discussed in Section V. Finally, concluding remarks are given in Section VI.

II. BRIEF REVIEW OF REORDERED NORMAL BASIS AND MULTIPLICATION OPERATION USING THIS BASIS IN $\mathbb{F}_{2^m}$

Let $2m+1$ be a prime number and β be a primitive $(2m+1)$th root of unity in $\mathbb{F}_{2^m}$, i.e., $\beta^{2m+1}=1$. Then, $\gamma=\beta+\beta^{-1}$ generates an ONB of type II in $\mathbb{F}_{2^m}$, which can be expressed as $I=\{\gamma^{2^i}, i=0,2,\ldots,m-1\}$. Define a set of m elements as $I_2=\{\gamma_i=\beta^i+\beta^{-i}, i=1,2,\ldots,m\}$. It has been proven in [4] that I_2 is also a basis in $\mathbb{F}_{2^m}$. In fact, it can be shown that I_2 is a permutation of the ONB I in the sense that it contains the same elements as I but in a different order. The basis $I_2=\{\gamma_1,\ldots,\gamma_m\}$ is referred to as the RNB.

Assume that A and B are two arbitrary elements in $\mathbb{F}_{2^m}$ and are represented with respect to the RNB basis $I_2=\{\gamma_1,\ldots,\gamma_m\}$ as

$$A=\sum_{i=1}^{m} a_i\gamma_i, \quad B=\sum_{i=1}^{m} b_i\gamma_i$$

and the product of the two elements is to be stored in C with respect to the same basis as $C=\sum_{i=1}^{m} c_i\gamma_i$, where a_i, b_i, $c_i\in F_2$. Following [15], the values for the product coordinates, c_i, can be calculated as:

$$c_i=\sum_{j=0}^{m} a_j[b_{s(i+j)}+b_{s(j-i)}], \quad i=1,\ldots,m \tag{1}$$

where function $s(i)$ facilitates the calculations by mapping the set of integers to the set $\{0,1,\ldots,2m+1\}$ and is defined in [15] as

$$s(i)\triangleq\begin{cases} i \bmod 2m+1, & 0\le i \bmod 2m+1\le m\\ 2m+1-i \bmod 2m+1, & \text{otherwise.}\end{cases}$$

III. REVIEW OF EXISTING RNB AND TYPE-II ONB MULTIPLIERS

A. Related Work

As mentioned, each element of an RNB also belongs to a corresponding ONB. In other words, since the multiplicative order of β is $2n+1$ and for each $0\le i\le n-1$, $\gamma^{2^i}=\beta^{2^i}+\beta^{2^{-i}}=\gamma_{2^i}$, there is an integer k such that $i\equiv\pm 2^k \bmod 2n+1$, and thus $\gamma_i=\gamma^{2^k}$ [4]. As a result, an RNB representation of a field element can be converted into type-II ONB representation by performing a simple permutation over the coordinates and vice versa. Therefore, RNB multipliers and type-II ONB multipliers can be used interchangeably without imposing hardware overhead. Consequently, it would be necessary to include type-II ONB multipliers when considering an RNB multiplier.

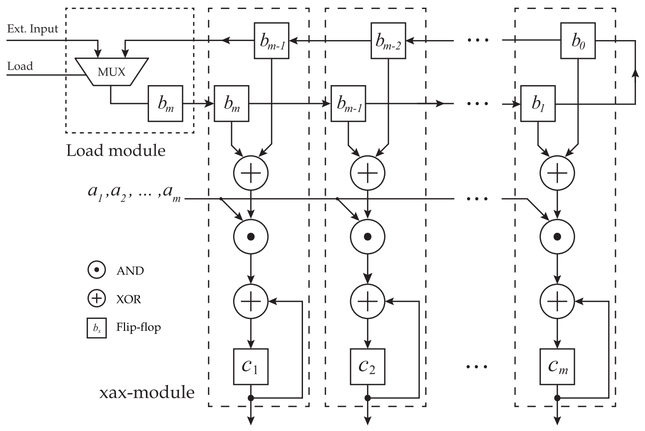

Fig. 1. SIPO RNB multiplier composed of xax-modules.

Several different architectures for sequential RNB/type-II ONB multipliers have been proposed in the literature. Table I gives a comparison between the hardware complexities of the multipliers in terms of critical path delay and the number of logic cells used. To draw a fair comparison, all of the chosen multipliers listed in the table require m clock cycles to generate a full set of product coordinates. The first two rows of the table show the well-known Massey–Omura NB multiplier [5] and its improved version proposed by Gao and Sobelman [6]. The next two architectures are presented by Agnew *et al.* [7] and Feng [8], respectively. The proceeding two rows list sequential multipliers with parallel output of types I and II [9] followed by two XOR efficient digit serial and AND efficient digit serial architectures proposed by Reyhani-Masoleh and Hasan [10]. The ninth and tenth rows show two efficient serial out/parallel out architectures for odd values of m [11] followed by Azarderakhsh's high-throughput PISO architecture [12]. Kwon *et al.* [13] and Yang *et al.* [14] also presented two other architectures for odd values of m. The last two rows in the table tabulate the SIPO and PISO RNB multipliers proposed by Wu *et al.* [15]. Note that some of the multipliers presented in Table I are Gaussian NB multipliers of type $k = 2$, which are essentially equal to type-II ONB multipliers.

B. Selected Multiplier Architecture in Reordered Normal Basis

This paper concentrates on the SIPO architecture proposed in [15] for four reasons.

1) As shown in Table I, this architecture possesses one of the smallest critical path delays in comparison to similar architectures.
2) The structure of this multiplier is highly regular, which makes it well suited for a full/semicustom VLSI implementation.
3) This architecture forms a basis to construct several world-level multipliers, such as those presented in [17]–[19].
4) The critical path of the architecture is independent of field size m. As a result, this architecture can be scaled to any arbitrary size multiplier without causing negative effects on the multiplication delay.

Fig. 1 shows the architecture of the SIPO multiplier presented in [15] to realize (1). As can be seen in the figure, this architecture is highly regular consisting of multiple copies of a building block referred to as an *xax-module*. This module, as shown inside the dashed boxes, consists of two XOR gates, one AND gate, and three flip-flops. The desired multiplier can be obtained by cascading the appropriate number of xax-modules. In the architecture at hand, the shift register at the top of the figure should be initially loaded with one of the input coordinates preceding the multiplication operation, whereas the other input is serially fed into the multiplier, one coefficient at a time, during the multiplication process. After the top shift register is fully loaded, a load signal cuts off the external data stream to the multiplexer, thus forming a circular shift register required for the multiplication process during the clock cycles that follow. After m clock cycles, the product coordinates, c_i, can be read from the output registers.

IV. Design of the Multiplier's Main Building Block in Domino Logic

As can be seen in Fig. 1, the xax-module contains the critical path of the SIPO architecture. This path is made of the two XOR gates in addition to the AND gate inside the xax-module. In an attempt to reduce the multiplication delay, Namin *et al.* [16] have designed and realized the critical path of the multiplier in domino logic. Although using domino logic for implementing the main building block of the multiplier can effectively increase the maximum operating frequency, this technique has a deteriorating effect on the power dissipation. The increase in power consumption stems from a higher internal switching activity, which is an inherent characteristic of domino logic circuits. Consequently, the resulting design would consume much more dynamic power compared to its static CMOS counterpart.

In order to alleviate the negative effects incurred by using domino logic-based designs over static CMOS, several techniques have been proposed in the past few years, e.g., high-speed domino [20], XOR-based domino [21], conditional-keeper domino [22], single-phase domino [23], and current comparison-based domino [24]. Primarily, the focus of the existing techniques is on design strategies, which compensate for leakage current in deep submicrometer technologies, narrower noise margins, contention delay at the evaluation phase, and the transistor stacking effect. Therefore, these techniques would be better suited for high fan-in circuits, in which the pull-down network (PDN) contains a large number of parallel paths to the ground, such as high fan-in multiplexers, comparators, and more general OR-like cells. Furthermore, the relatively large number of transistors required to implement these techniques compared to the total number of transistors used in the design of the small xax-module imposes significant power and area overheads. Such techniques are not applicable to the multiplier in discussion.

The implementation presented in [16] uses a conventional domino logic circuitry, thus increasing the dynamic power consumption due to the higher switching activities. In this paper, the power dissipation problem is tackled using custom-designed domino circuitry, which reduces the contention cur-

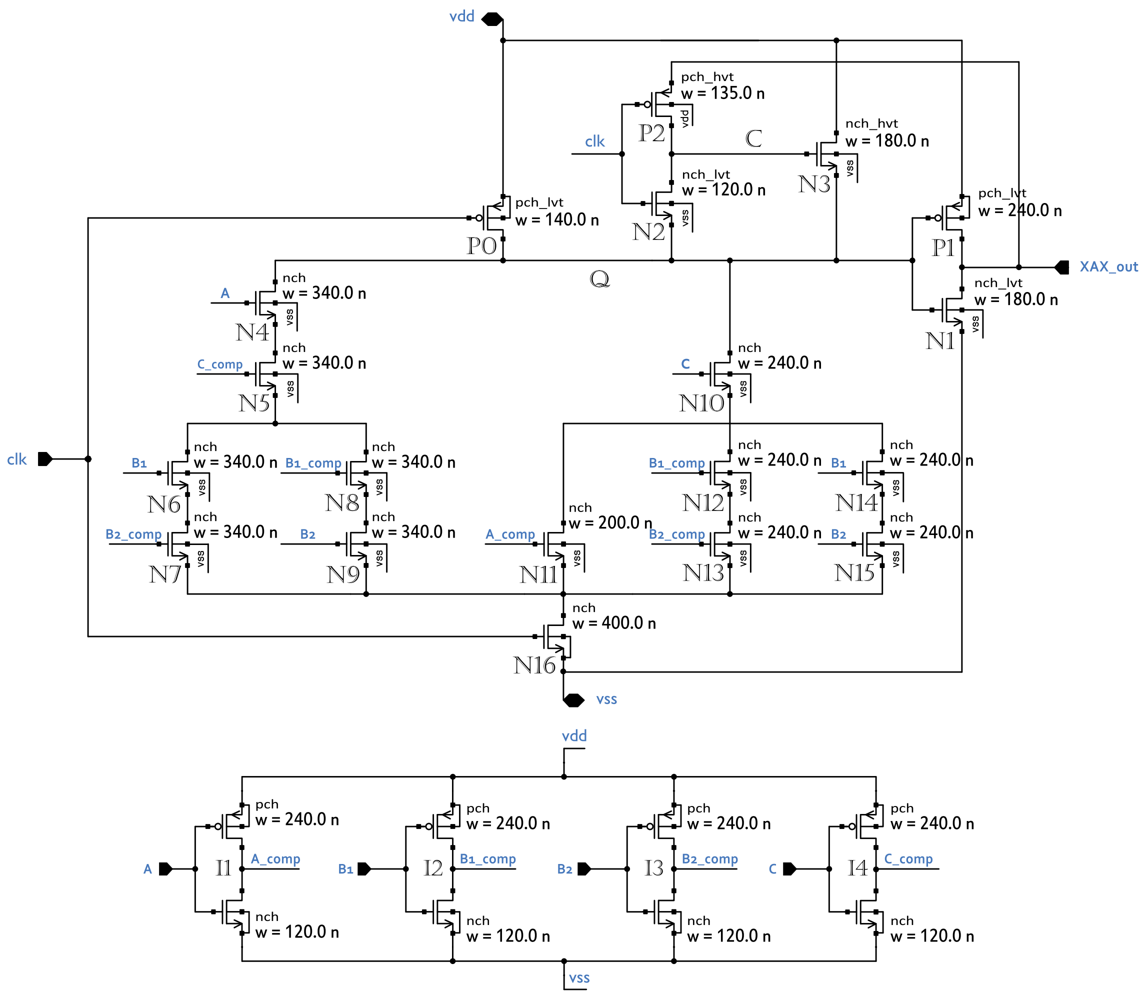

Fig. 2. Proposed design for XOR–AND–XOR function in domino logic.

rent drawn at the very beginning of the evaluation phase. Depending on the value of the input signals, contention may occur between the pull-up network (PUN) and the PDN of a domino circuit during the evaluation phase. This contention, though short in time, forms a conducting path from V_{DD}, across PUN and PDN, to ground causing high-amplitude current spikes. The basic idea is to limit the contention current by utilizing a new conditional keeper to compensate for the power overhead caused by the higher switching activity of the circuit.

Fig. 2 shows a schematic of the circuit designed to implement the XOR–AND–XOR function in domino logic. This circuit is responsible to realize an XOR operation between two different coordinates of B, followed by an AND applied to the result and one of the A coordinates. Finally, this is combined with another XOR that, when paired with a flip-flop, forms an accumulation unit. In terms of the variables used in (1), this circuit realizes logic function $((b_1 \oplus b_2)\ .\ a) \oplus c$. The static PUN is merely composed of a single pMOS transistor charging the dynamic node Q to V_{DD} during the precharge phase. The PDN, on the other hand, consists of 12 transistors (N4–N15), which discharge the dynamic node at the presence of appropriate combinations of the input values. The PDN is connected to a footer transistor, N16, which reduces the leakage current due to the stacking effect and opens a path to the ground during the evaluation phase. Transistors P2 and N2 generate a control signal to an nMOS keeper depending on the voltage of the dynamic node and the logic state of the clock signal. Transistors P1 and N1 form the output inverting stage, providing the required current to drive the output flip-flop. In the presented schematic, the input signals are referred to as *B1*, *B2*, *A*, and *C*. Four inverter gates shown at the bottom of Fig. 2 (I1–I4) generate the complements of the module's input signals. As a naming convention, a "*_comp*" added to the end of the signal's name refers to its complement signal. The proposed dynamic circuit operates in two phases as follows.

During the precharge phase, pull-up transistor P0 steadily charges the dynamic node. If the dynamic node is initially in a low state, node C is quickly charged to V_{DD} by P2, which turns ON the keeper transistor to speed up the precharging process. The voltage of the dynamic node rises until it reaches a certain level, at which time the output switches to a low state, causing P2 to discharge node C and then turn OFF the keeper transistor. Therefore, at the end of the precharge phase, the dynamic node is fully charged and the keeper is held OFF to avoid negative impacts on delay and power consumption at the beginning of the next phase.

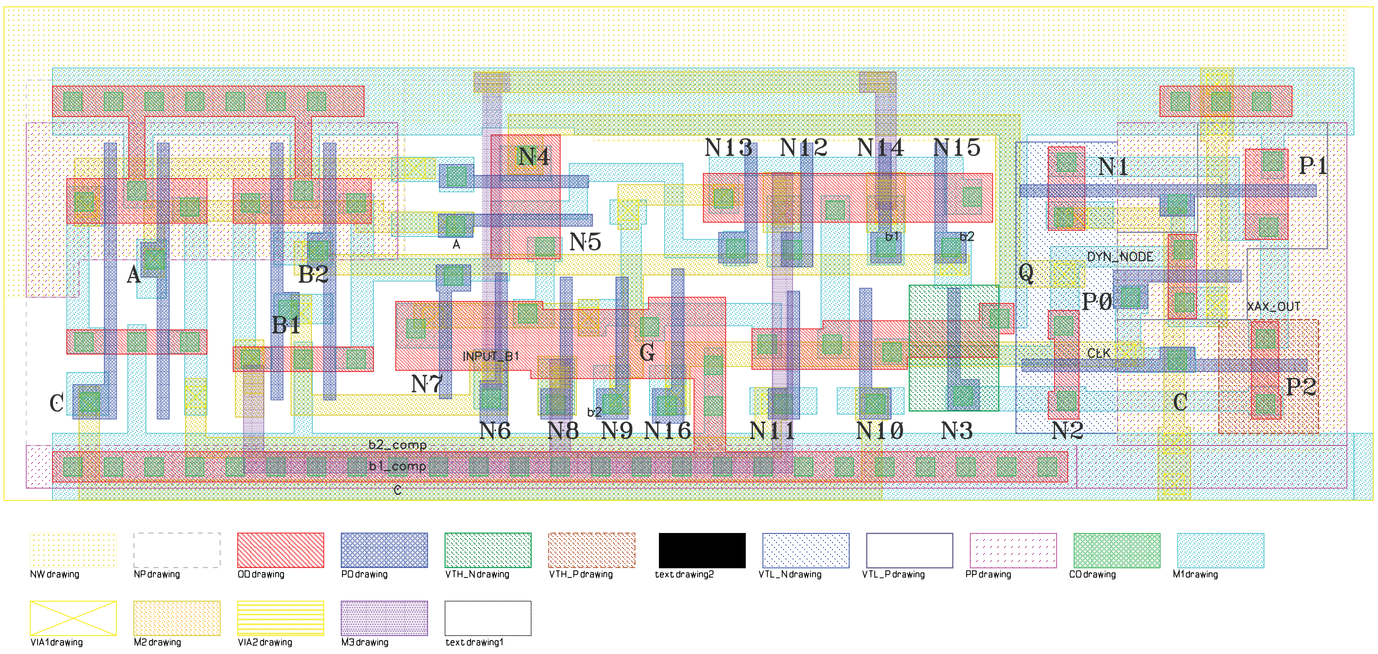

Fig. 3. Layout for the XOR–AND–XOR function in 65-nm technology.

At the beginning of the evaluation phase, the clock signal switches to a high state, keeping the pull-up transistor turned OFF. At this moment, two different scenarios could occur depending on the logic values of the input signals. In the first scenario, a conducting path is formed from the dynamic node to the ground, discharging the dynamic node through the PDN network. In this case, when the dynamic voltage falls below $V_{\mathrm{DD}} - V_{\mathrm{th},N2}$, the source and drain junctions of transistor N3 are reversed and the accumulated charge on node C is fully discharged through N3. This prevents the keeper transistor from being turned ON. In the second scenario, the dynamic node is evaluated to a high state. N2 is turned ON in the case that the leakage current reduces the voltage of the dynamic node. The behavior of the circuit shown in Fig. 2 is explained in more detail in our recent work in [25].

V. Custom-Layout Implementation of the xax-Module

After determining the optimal values of the transistor sizes through extensive schematic-level simulations, the layout of the schematic shown in Fig. 2 was created in a seven-metal 65-nm CMOS process. A large number of major and minor modifications were applied to the prototype layout in an attempt to find the best transistor arrangement to reduce the required area and to find the best routing path patterns. Post-layout fine-tuning was also carried out to reduce the negative effects of parasitic components, which, in turn, resulted in less delay. The final layout is shown in Fig. 3. The leftmost section shows four inverters used to generate the complements of the input signals. The section in the middle is responsible for implementing the PDN. Finally, the rightmost section contains the keeper transistor, the keeper control circuit, and the output inverter. The dimensions of the layout are 1.8 μm × 6.37 μm covering 11.467 $\mu\mathrm{m}^2$. The center-to-center power rail pitch was selected, such that the layout would be compatible with

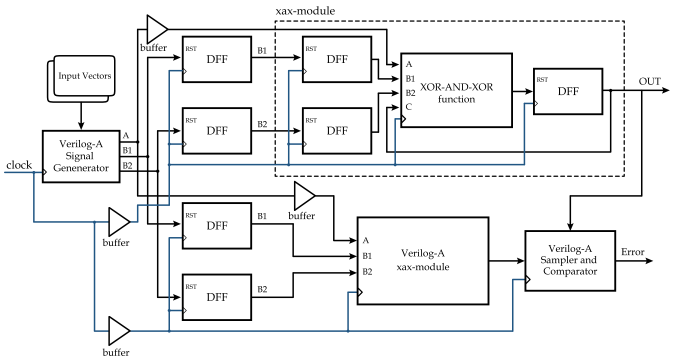

Fig. 4. Test bench for performing corner analysis.

the 65-nm general purpose standard cell library from TSMC, thus allowing it to share the same size power rails.

The final layout of the xax-module also includes two input and one output flip-flops. Since the logic function is realized by the PDN (as opposed to the case in which the function is realized by pMOS transistors in PUN), the use of negative-edge triggered flip-flops would be the only available option to provide enough time for the domino cell to evaluate. As mentioned earlier, the static pull-up transistor charges the dynamic node right after the falling edge of clock signal despite the values of the inputs, hence, causing an invalid output at the beginning of the precharge phase. As a result, the output flip-flop is required to have a hold-time equal or preferably less than zero to be able to sample the evaluated output at the right moment.

A set of Calibre tools was used for debugging and verifying the physical layout. Physical verification and mask set corrections were carried out via the Calibre DRC tool. Circuit verification was performed by making device and connectivity comparisons between the physical layout and the schematic through Calibre LVS. Finally, parasitic information was extracted using Calibre PEX for accurate postlayout analysis and simulation. Fig. 4 shows the test-bench scheme used to verify the correct functionality of the xax-module and to evaluate the performance of the module. A set of random input vectors was first generated using *C* programming language. At each clock cycle, the Verilog-A module reads a new vector from the input file and converts the digital values to their corresponding analog signals. In order to account for loading capacitances, fall time and rise time, a small low-pass filter was defined at the final stage of the signal generator. To achieve more compatibility in terms of delay, the signal generator interfaces the xax-module through a set of standard flip-flops generating more realistic signals. The same set of input signals is also fed into a Verilog-A module implementing the same functional behavior of the xax-module. The outputs of the two modules are compared against each other at each clock cycle and an error pulse is generated in the case of mismatch.

In order to measure the amount of deviations in power consumption and maximum operating frequency over a range of process, temperature, and voltage variations, various simulations were conducted. We first performed corner analysis at three different conditions: room temperature of 27 °C and two extreme temperatures of −30 °C and +85 °C. All of the simulations were done with a supply voltage of 1.2 V. The results are listed in Table II. As expected, the worst case delay occurs in the slow–slow corner at +85 °C (129 ps), whereas the best-case delay belongs to the fast–fast corner at −30 °C (79 ps). The amount of power consumption ranges from 31.8 μW/GHz for the slow–slow corner at −30 °C to 41 μW/GHz for the fast–fast corner at +85 °C. Table II presents the results of corner analysis for cases in which temperature is 27 °C and V_{DD} varies between 1.08 V (−10%) and 1.32 V (+10%) for each corner. As the simulation results suggest, the proposed implementation demonstrates acceptable performances over a wide range of temperatures, voltages, and process variations.

TABLE II

POSTLAYOUT CORNER ANALYSIS SIMULATION RESULTS OF THE xax-MODULE; DELAY (PS) AND POWER (μW/GHz). (a) DELAY AND POWER VERSUS TEMPERATURE AT $V_{\mathrm{DD}} = 1.20$ v. (b) DELAY AND POWER VERSUS VOLTAGE AT 27 °C

(a)

Corner		**Temperature** (°C)		
		-30	**+27**	**+85**
Fast-Fast	Delay	79	81	83
	Power	37.38	38.97	41.00
Fast-Slow	Delay	93	97	101
	Power	33.95	35.28	36.72
Typ./Typ.	Delay	89	95	100
	Power	34.51	35.49	37.26
Slow-Fast	Delay	89	96	103
	Power	34.39	35.57	36.93
Slow-Slow	Delay	120	125	129
	Power	31.8	33.02	34.82

(b)

Corner		**Voltage (V)**			
		-10% $1.08v$	-5% $1.14v$	$+5\%$ $1.26v$	$+10\%$ $+1.32v$
Fast-Fast	Delay	86	84	81	80
	Power	34.31	36.35	41.86	44.71
Fast-Slow	Delay	108	101	88	85
	Power	30.82	33.50	38.06	40.54
Typ./Typ.	Delay	108	99	89	86
	Power	90.95	33.15	38.07	40.49
Slow-Fast	Delay	107	101	90	87
	Power	31.19	33.43	42.93	40.31
Slow-Slow	Delay	143	119	110	105
	Power	29.28	31.01	34.21	37.77

VI. DESIGN AND IMPLEMENTATION OF THE FULL MULTIPLIER USING THE xax-MODULE

As shown in Fig. 1, a full multiplier of an arbitrary size can be readily constructed by serially connecting xax-modules together. In the context of cryptography, a proper choice of field size depends on several factors, including the features of the chosen representation basis and the level of security required. It has been shown that a 160-bit ECC can provide the same level of security as a 1024-bit RSA security scheme, which provides adequate security for a broad range of applications [26]. In order to implement the full multiplier, we selected the field size of 233, as it can be represented by an RNB and it is large enough to fall in the suitable range for ECC. Additionally, it is one of the few fields recommended by the National Institute of Standards and Technology (NIST) for ECC applications.

The arrangement and interconnections between different building blocks of the 233-bit multiplier are shown in Fig. 5. To achieve the most compact design, the chain of the xax-modules was broken down into an array of 18 rows, each containing 13 modules. In each row, three buffers were used to provide enough driving strength for the high fan-out $input\_a$, $clock$, and $reset$ signals. These buffers, in turn, were driven by additional buffers, thus resulting in a buffer chain shown at the leftmost part of the figure. To further reduce the dynamic power dissipation caused by the domino circuit, clock-gating was used as a complimentary technique. At the beginning of each multiplication operation, the domino circuit is idle, while the coordinates of one of the inputs are being loaded into the input shift register. However, the dynamic nodes in the xax-modules are frequently charged and discharged by PUD and PDN causing unnecessary power overhead. This can be avoided by keeping the clock signals entering the domino circuits separate, such that they may be disabled during the load process.

After the coordinates are fully loaded into the shift register, the external input to the first flip-flop should be replaced by the output of the last flip-flop to form a circular shift register. This can be done by a multiplexer, which uses an input selector to choose between the external input and the output of the last flip-flop. The critical path of a typical multiplexer is composed of one inverter, one AND gate, and one OR gate. This multiplexer can potentially become a delay bottleneck

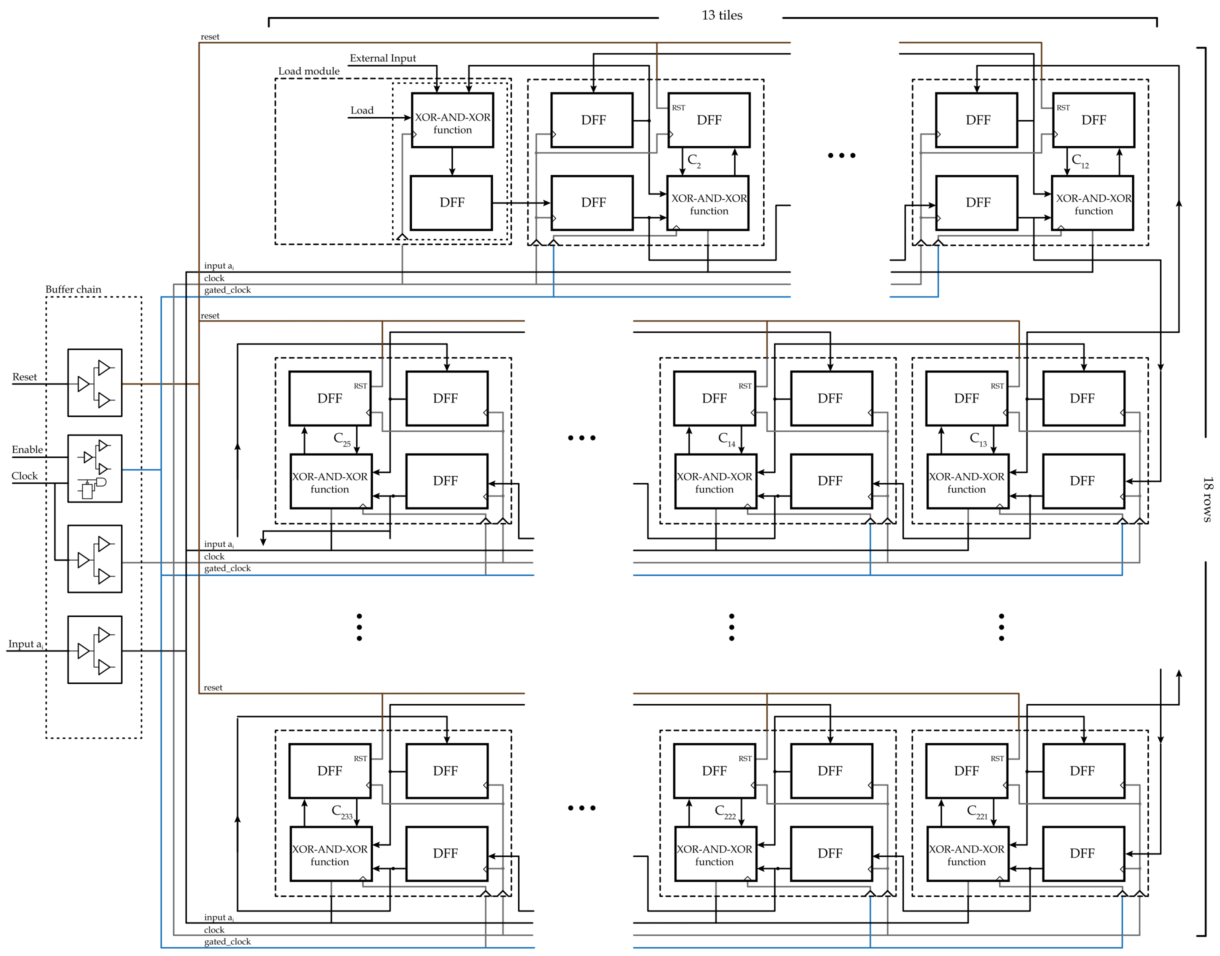

Fig. 5. Block diagram of a full 233-bit RNB multiplier.

if its critical path in CMOS implementation becomes longer than the delay of an XOR–AND–XOR cell. Adopted from the implementation in [16], if connected properly, an XOR–AND–XOR circuit may be used in lieu of a multiplexer. While ensuring consistency across the design, this idea can prevent the multiplexer from being a delay bottleneck by realizing it in domino logic. Assuming that input $B1$ is grounded, input A is chosen as the input selector, and finally, C and $B2$ are connected to the external input and the output of the last flip-flop, respectively. When the *selector* signal is at a low state, the output would be evaluated as $(('0' \oplus flop\_out) \cdot '0')\oplus external\_input$, which is equal to *external_input*. When the *selector* signal is at a high state and *external_input* is held low, the output would equate to $(('0'\oplus flop\_out) \cdot '1')\oplus '0' = flop\_out$. This special module, which is essentially one half of an xax-module, is referred to as the *Load module* and is located at the beginning of the first row of Fig. 5.

Fig. 6 shows the final layout of the 233-bit multiplier implemented in a CMOS 65-nm process using a Cadences Layout Composer. The buffer chains, Load module, and one of the xax-modules are highlighted in the figure. The dimensions of the full layout are 171 μm $\times$ 71 μm for a total area of 12 141 μm^2, which includes the outer power rings. Without considering the power rings, the area utilization was measured to be 10 743 μm^2.

VII. Simulation Results and Performance Comparison Between Different VLSI Implementations

This section draws a comparison between the characteristics of the proposed VLSI implementation and those of several implementations reported in the literature. To perform accurate simulations, the parasitic information of the full multiplier was extracted using Calibre PEX. At this phase, 735 532 different components, including parasitic capacitances and resistances, were extracted from the physical layout. In the next phase, the simulations were performed in Cadences Analog Environment using the Spectre simulator to measure the power consumption and the maximum operating frequency of the circuit. To ensure the correct functionality of the circuit, a presimulation stage was required, in which the test data set was generated. To do so, the functional behavior of the multiplier was also modeled in MATLAB. Then, a large array of random 233-bit paired vectors were created and fed into the MATLAB code to generate a set of *golden* product coordinates. Input pairs and their corresponding outputs were stored in two separate files. During the analog simulation, a Verilog-A module read the input files and fed an input pair into the multiplier for each multiplication operation. After each multiplication operation, the output coordinates were sampled

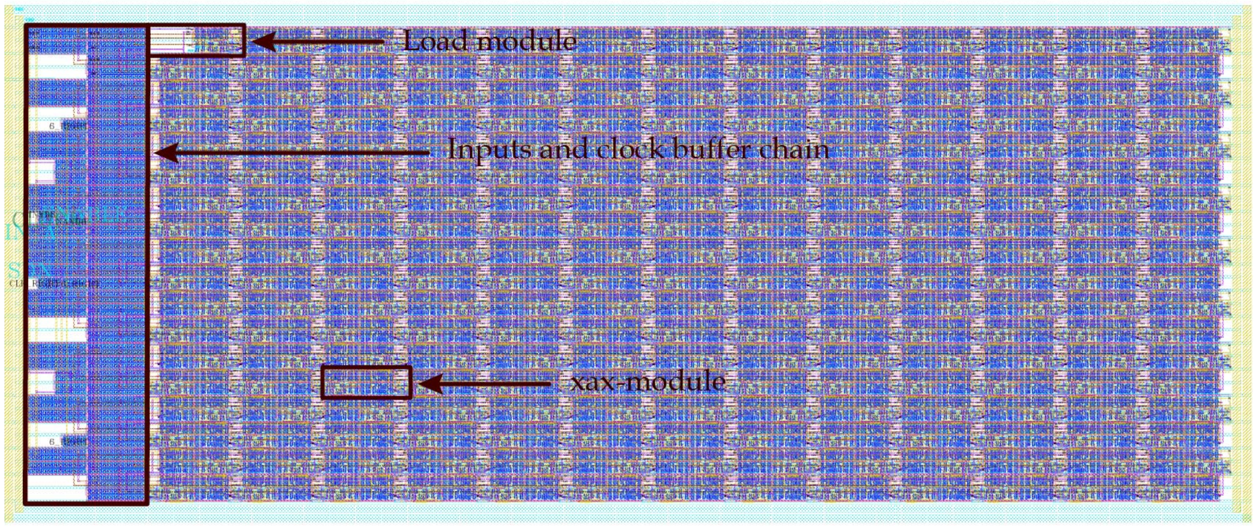

Fig. 6. Proposed layout for a 233-bit sequential RNB multiplier designed in domino logic.

and stored in an output file before new data were loaded into the multiplier. These outputs were later verified by comparing them against the golden set created by the MATLAB code. The simulation result showed that the circuit was correctly functional up to a clock rate of 3.84 GHz. The power consumption of the multiplier was measured to be 13.01 mW/GHz averaged over 100 consecutive multiplication operations.

As previously emphasized in Section I, the main objective of this paper was to compare the performance of the proposed implementation with that of a static CMOS implementation to demonstrate that the new domino logic circuit can further reduce the multiplication delay of the multiplier while preserving the total power consumption. To achieve a fair and accurate comparison, we also implemented the layout of the static CMOS multiplier in the same 65-nm CMOS process using standard cells from TSMC's libraries. The layout of the static design was constructed based on the same structure shown in Fig. 5. Note that the Load module was implemented in static CMOS and then was incorporated in the layout to provide the same functionality as its counterpart in domino logic. To ensure consistency, the same set of random inputs was applied to the static multiplier when conducting the simulations. This realization has a maximum operating frequency of 2.94 GHz and requires 238 ns to finish a single multiplication operation. Including the power rings, the size of the layout is 153 μm $\times$ 71 μm equal to an area of 10 863 μm^2. The required area is reduced to 9 574 μm^2 when not considering the outer rings.

Table III summarizes the main characteristics of the two custom-layout implementations. The column entitled "Critical Path Delay" corresponds to the maximum operating frequency of the design, whereas "Multiplication Delay" indicates the amount of time required to perform one multiplication operation. As can be seen in the table, compared to the static CMOS design, the new design has a considerably smaller critical path delay, i.e., 260 versus 340 ps, equal to 31% improvement in the maximum operating frequency. Also, employing the proposed domino circuit has successfully decreased the power dissipation of the domino implementation. In fact, the power dissipation achieved is even marginally less than that of the static CMOS design. It is generally expected that a domino circuit requires a fewer number of transistors compared to its corresponding static CMOS circuit, as the logic function is realized in PDN using only nMOS transistors. However, in the case of the domino multiplier, strict limitations on the cell's height, isolated oxide areas, the presence of two n-well islands, and requirements of the design rules in 65-nm process regarding the use of multithreshold transistors resulted in a moderate increase in the area of the xax-module. The full design shown in Fig. 6 requires about 11% more area than the static design counterpart.

In recent years, many efforts have been made to design efficient finite field multipliers at the algorithmic level; however, not many VLSI implementations have been reported in the open literature. Moreover, the presence of different conditions and assumptions in each implementation, such as different field sizes, diverse ranges of semiconductor technologies, different representation systems, varying levels of parallelism used, different levels of implementations, and so on, makes it difficult to draw a fair comparison between the performances of these various implementations.

Table IV lists the specifications and hardware complexities of the proposed implementation along with those of published implementations. The last column of the table shows the levels of implementation for which the results have been reported. As can be seen in the table, not all of the implementations have been performed at the same level. A postsynthesis implementation is a netlist-level implementation, which is carried out under more idealistic conditions. At this level of implementation, the negative effects of parasitic components, clock tree buffers, high fan-out nets, and routing paths are not taken into account, leading to more optimistic results. To get a broader picture of the potential differences between the results of these two levels of implementation, the results

TABLE III
COMPLEXITY COMPARISON BETWEEN VLSI IMPLEMENTATIONS OF DOMINO LOGIC AND STATIC CMOS DESIGNS FOR A SEQUENTIAL RNB MULTIPLIER IN GF(2^{233})

Architecture	Field size	Critical Path Delay (nm)	Multiplication Delay (nm)	Power (mW/GHz)	Area (μm^2)	Technology
static CMOS	233	0.340	238	13.48	$10,863$	$65nm$
Proposed	233	0.260	182	13.01	$12,141$	$65nm$

TABLE IV
COMPLEXITY COMPARISON OF DIFFERENT VLSI IMPLEMENTATIONS FOR FINITE FIELD MULTIPLIERS OVER GF (2^m)

Architecture	Base	Field size	CPD* (ns)	Delay** (ns)	Power (mW/GHz)	Area (μm^2)	Area×Delay	Technology (nm)	Implementation Level
Agnew [27]	RNB/ONBII	155	25	3900	—	—	—	2000	Synthesis
Tang [28]	RNB/ONBII	233	7.69	230.7	184.4	$189,297$	$43,670,817$	180	Place&Route
Satoh [29]	Poly	160	1.96	256.75	—	—	—	130	Synthesis
Ansari [30]	Poly	163	8.0	40	—	$1,272,102$	$50,884,080$	180	Place&Route
Azarderakhsh (SIPO) [12]	GNB	163	0.93	13.95	—	$34,278$	$478,178$	65	Synthesis
Azarderakhsh (PISO) [12]	GNB	163	1.38	22.70	—	$34,837$	$790,800$	65	Synthesis
Hariri [31]	SPB	131	0.28	67	836.67	$752,009$	$50,384,603$	65	Synthesis
Wang (Kwon) [32]	RNB/ONBII	233	0.29	67.86	—	$2,581,876$	$175,206,105$	65	Synthesis
Wang [32]	GNB	233	0.19	22.61	—	$1,105,295$	$24,990,720$	65	Synthesis
Namin [16]	RNB/ONBII	233	0.630	293.58	85.1	$109,644$	$32,189,286$	180	Place&Route
Leboeuf [33]	RNB/ONBII	233	0.559	260.5	83.7	$62,048$	$16,163,504$	180	Place&Route
Rashidi [34]	GNB	226	1.525	344.65	—	$48,076$	$16,569,393$	180	Place&Route
Hashemi [35]	Poly	233	2.19	65.7	7.69	$14,323$	$941,021$	65	Synthesis
static CMOS	RNB/ONBII	233	0.340	238	13.48	$10,863$	$2,585,394$	65	Place&Route
Proposed	RNB/ONBII	233	0.260	182	13.01	$12,141$	$2,209,662$	65	Place&Route
Proposed (Syn.)	RNB/ONBII	233	0.155	108.5	9.59	$9,601$	$1,041,708$	65	Synthesis

* Critical Path Delay.
** Time delay required to perform a single multiplication operation.

of the proposed implementation at the schematic level are also presented in the last row of the table. Note that the schematic level simulation is, in essence, an analog simulation. However, since our schematic implementation only incorporates standard cells and transistors, it falls into the same level as synthesis implementation. It is important to note that compared to the postlayout implementation, the last row shows about 26% and 40% decrease in power consumption and multiplication delay, respectively.

If all the implementations listed in Table IV were sequential multipliers, multiplication delay would be a convenient measure of evaluation. However, some of the entries of Table IV utilize different levels of parallelism in their architectures; this enables them to complete one multiplication operation in fewer clock cycles. On the other hand, this has a significant effect on the area requirement and power dissipation of the multiplier. For each higher level of parallelism that is used, there will be a shorter multiplication delay and the multiplier will become progressively bigger. The product of delay and area can appropriately reflect both factors simultaneously and can be used as a good measure of evaluation.

At the synthesis level, most of the multipliers in the list have a lower multiplication delay than the proposal. However, they all belong to the class of digit-level multipliers, which are well reflected in the area requirements of the multipliers. For example, Wang and Fan's [32] implementation requires over 115 times more area and both Azarderakhsh and Reyhani-Masoleh's [12] designs are almost 3.5 times bigger than the proposal. The proposed implementation is the smallest in the list, requiring 33% less area than the second smallest implementation in the table. The results presented in the "Delay × Area" column suggest that the implementation being proposed has the fourth lowest complexity, after both Azarderakhsh and Reyhani-Masoleh's [12] implementations and Namin *et al.*'s [35] realization. However, in all the three cases, the preloading delay associated with the input registers has been excluded from the multiplication delay, which would add another m cycles ($m \times CPD$) to latency. Adopting the same approach in our calculations would decrease the multiplication delay by a factor of three. Furthermore, it should be taken into account that Azarderakhsh and Reyhani-Masoleh's [12] multiplier uses a smaller field size (163 instead of 233 for the proposed), which has a direct effect on both multiplication delay and area requirement. At this level of implementation, all of the entries were realized in 65-nm technology, aside from the first two.

At the layout level, all implementations, excluding our static and domino designs, were realized in technology nodes older than 65 nm. Ansari and Wu's [30] design is the only implementation with shorter multiplication delay than ours.

Considering the fact that its critical path is the longest among all of the implementations listed in Table IV (excluding Agnew), the shorter multiplication delay was achieved by limiting the multiplication operation to mere five clock cycles at the cost of significantly greater area requirements. This is well reflected in the area and delay-area quantities. For example, the area requirement of this implementation is more than 100 times larger than the proposal. Showing the smallest area $\times$ delay index, the proposed design compares favorably to all other multipliers at this level of implementation. Finally, considering the product of area and delay as a measure of evaluation, the domino-based implementation outperforms the static implementation by a difference of about 15%.

Although the information presented in Table III provides a broad picture of the performance of finite field multipliers, it is important to note that due to the limited number of reported implementations, it may not be easy to draw a fair and accurate comparison yet. As the dimensions of semiconductors decrease, the negative effect of interconnect and parasitic capacitances becomes more dominant in contributing to the capacitive load on chips. Consequently, the gap between schematic/synthesis-level and postlayout simulations grows bigger. This makes it more difficult to form an accurate estimate of the overall performance of a multiplier before the negative effects of routing paths and parasitic capacitance are taken into consideration. The limited number of reported implementations and the presence of different conditions and assumptions in each case suggest that more attention should be devoted to layout-level implementations of finite field multipliers to enable more comprehensive and accurate comparisons.

VIII. Conclusion

A new VLSI implementation of a 233-bit SIPO finite field multiplier was presented. The field size of 233 is currently recommended by the NIST for embedded security applications using ECC. The proposed design is highly regular, possessing a repeating pattern of a single building block implemented in domino logic, which can be readily scaled to any arbitrary size multiplier by cascading the appropriate number of blocks. In an attempt to alleviate the high-power dissipation of the domino circuit stemming from higher internal switching activities, the original design of this building block was modified to reduce the contention current drawn at the very beginning of the evaluation phase. The post place-and-route simulations showed the correct functionality of the design up to a clock range of 3.85 GHz, achieving a much higher operating speed while consuming marginally less power compared to the static CMOS counterpart. The same design methodology can be utilized to improve the operating speed of other similar regular architectures without compromising power consumption.

References

[1] *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Standard 1363-2000, Aug. 2000, pp. 1–228.

[2] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, 2nd ed. New York, NY, USA: Cambridge, U.K.: Cambridge Univ. Press, 1997.

[3] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, "Optimal normal bases in GF(p^{n})," *Discrete Appl. Math.*, vol. 22, no. 2, pp. 149–161, Feb. 1989.

[4] S. Gao and S. A. Vanstone, "On orders of optimal normal basis generators," *Math. Comput.*, vol. 64, no. 211, pp. 1227–1233, 1995.

[5] J. K. Omura and J. L. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent 4 587 627, May 6, 1986.

[6] L. Gao and G. E. Sobelman, "Improved VLSI designs for multiplication and inversion in $GF(2^m)$ over normal bases," in *Proc. 13th Annu. IEEE Int. ASIC/SOC Conf.*, Sep. 2000, pp. 97–101.

[7] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone, "An implementation for a fast public-key cryptosystem," *J. Cryptol.*, vol. 3, no. 2, pp. 63–79, Jan. 1991.

[8] G.-L. Feng, "A VLSI architecture for fast inversion in $GF(2^m)$," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, Oct. 1989.

[9] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity word-level sequential normal basis multipliers," *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 98–110, Feb. 2005.

[10] A. Reyhani-Masoleh and M. A. Hasan, "Efficient digit-serial normal basis multipliers over $GF(2^m)$," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 5, May 2002, pp. V-781–V-784.

[11] A. Reyhani-Masoleh, "Efficient algorithms and architectures for field multiplication using Gaussian normal bases," *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 34–47, Jan. 2006.

[12] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 744–757, Apr. 2013.

[13] S. Kwon, K. Gaj, C. H. Kim, and C. P. Hong, "Efficient linear array for multiplication in $GF(2^m)$ using a normal basis for elliptic curve cryptography," in *Cryptographic Hardware and Embedded Systems*, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 76–91.

[14] D. J. Yang, C. H. Kim, Y. Park, Y. Kim, and J. Lim, "Modified sequential normal basis multipliers for type II optimal normal bases," in *Computational Science and Its Applications*. Berlin, Germany: Springer-Verlag, 2005, pp. 647–656.

[15] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao, "Finite field multiplier using redundant representation," *IEEE Trans. Comput.*, vol. 51, no. 11, pp. 1306–1316, Nov. 2002.

[16] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "High-speed hardware implementation of a serial-in parallel-out finite field multiplier using reordered normal basis," *IET Circuits, Devices Syst.*, vol. 4, no. 2, pp. 168–179, Mar. 2010.

[17] A. H. Namin, H. Wu, and M. Ahmadi, "Comb architectures for finite field multiplication in $\mathbb{F}_2^m$," *IEEE Trans. Comput.*, vol. 56, no. 7, pp. 909–916, Jul. 2007.

[18] A. H. Namin, H. Wu, and M. Ahmadi, "A high-speed word level finite field multiplier in $\mathbb{F}_2^m$ using redundant representation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 10, pp. 1546–1550, Oct. 2009.

[19] A. H. Namin, H. Wu, and M. Ahmadi, "High-speed architectures for multiplication using reordered normal basis," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 164–172, Feb. 2012.

[20] M. H. Anis, M. W. Allam, and M. I. Elmasry, "Energy-efficient noise-tolerant dynamic styles for scaled-down CMOS and MTCMOS technologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 2, pp. 71–78, Apr. 2002.

[21] C.-H. Hua, W. Hwang, and C.-K. Chen, "Noise-tolerant XOR-based conditional keeper for high fan-in dynamic circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, May 2005, pp. 444–447.

[22] A. Alvandpour, R. K. Krishnamurthy, K. Soumyanath, and S. Y. Borkar, "A sub-130-nm conditional keeper technique," *IEEE J. Solid-State Circuits*, vol. 37, no. 5, pp. 633–638, May 2002.

[23] C. J. Akl and M. A. Bayoumi, "Single-phase SP-domino: A limited-switching dynamic circuit technique for low-power wide fan-in logic gates," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 2, pp. 141–145, Feb. 2008.

[24] A. Peiravi and M. Asyaei, "Robust low leakage controlled keeper by current-comparison domino for wide fan-in gates," *Integr., VLSI J.*, vol. 45, no. 1, pp. 22–32, 2012.

[25] P. H. Namin, R. Muscedere, and M. Ahmadi, "Low power design of a word-level finite field multiplier using reordered normal basis," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 437–440.

[26] *An Elliptic Curve Cryptography (ECC) Primer: Why ECC is the Next Generation of Public Key Cryptography. The Certicom 'Catch the Curve' White Paper Series*, Jun. 2004. [Online]. Available: https://www.certicom.com/content/dam/certicom/images/pdfs/WP-ECCprimer.pdf

[27] G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An implementation of elliptic curve cryptosystems over $F_{2^{155}}$," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 5, pp. 804–813, Jun. 1993.

[28] W. Tang, H. Wu, and M. Ahmadi, "VLSI implementation of bit-parallel word-serial multiplier in $GF(2^m)$," in *Proc. 3rd Int. IEEE-NEWCAS Conf.*, Jun. 2005, pp. 399–402.

[29] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Trans. Comput.*, vol. 52, no. 4, pp. 449–460, Apr. 2003.

[30] B. Ansari and H. Wu, "Efficient finite field processor for $GF(2^{163})$ and its VLSI implementation," in *Proc. 4th Int. Conf. Inf. Technol. (ITNG)*, Apr. 2007, pp. 1021–1026.

[31] A. Hariri and A. Reyhani-Masoleh, "Digit-level semi-systolic and systolic structures for the shifted polynomial basis multiplication over binary extension fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 11, pp. 2125–2129, Nov. 2011.

[32] Z. Wang and S. Fan, "Efficient Montgomery-based semi-systolic multiplier for even-type GNB of $GF(2^m)$," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 415–419, Mar. 2012.

[33] K. Leboeuf, A. H. Namin, H. Wu, R. Muscedere, and M. Ahmadi, "Efficient VLSI implementation of a finite field multiplier using reordered normal basis," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 1218–1221.

[34] B. Rashidi, S. M. Sayedi, and R. R. Farashahi, "An efficient and high-speed VLSI implementation of optimal normal basis multiplication over $GF(2^m)$," *Integr., VLSI J.*, vol. 55, pp. 138–154, Sep. 2016.

[35] S. H. Namin, H. Wu, and M. Ahmadi, "Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 441–449, Feb. 2017.

Parham Hosseinzadeh Namin was born in 1983 in Tehran, Iran. He received the B.Sc. degree in electrical engineering from Islamic Azad University, Karaj Branch, Karaj, Iran, in 2006, the M.Sc. degree in telecommunication engineering from the University of Tabriz, Tabriz, Iran, in 2009, and the Ph.D. degree in electrical engineering from the University of Windsor, Windsor, ON, Canada, in 2016.

He is currently a Senior Design Engineer with the SoC RTG Team, Advanced Macro Devices. His current research interests include digital and analog integrated circuits, architectures in finite fields, hardware implementation of cryptosystems, and hardware trojan detection/prevention techniques in VLSI.

Crystal Roma was born in Windsor, ON, Canada, in 1995. She received the B.A.Sc. degree in electrical engineering with a minor in mathematics from the University of Windsor, Windsor, in 2017. She is currently working toward the M.A.Sc. degree at the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, with a specialization in computer hardware.

Her current research interests include computer arithmetic and hardware realizations of cryptosystems.

Roberto Muscedere was born in Windsor, ON, Canada, in 1973. He received the B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of Windsor, Windsor, in 1996, 1999, and 2003, respectively.

During his studies, he also managed the microelectronics computing environment at the Research Center for Integrated Microsystems (formally VLSI Research Group), University of Windsor. He is currently an Associate Professor with the Electrical and Computer Engineering Department, University of Windsor. His current research interests include the implementation of high-performance and low-power VLSI circuits, full and semicustom VLSI design, computer arithmetic, HDL synthesis, and digital signal processing.

Majid Ahmadi (F'02) received the B.Sc. degree in electrical engineering from Sharif University, Tehran, Iran, in 1971, and the Ph.D. degree in electrical engineering from the Imperial College of Science, Technology and Medicine, London, U.K., in 1977.

He has been with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, since 1980, where he is currently a Distinguished University Professor and the Associate Dean of Engineering for Research and Graduate Studies. He has coauthored *Digital Filtering in 1-D and 2-Dimensions; Design and Applications* (New York: Plennum, 1989). His current research interests include digital signal processing, machine vision, pattern recognition, neural network architectures, applications, VLSI implementation, computer arithmetic, and MEMS. He has published over 500 articles in these areas.

Dr. Ahmadi is a fellow of the IET, U.K. He was a recipient of the Honorable Mention Award from the Editorial Board of the *Journal of Pattern Recognition* in 1992 and the Distinctive Contributed Paper Award from the Multiple-Valued Logic Conference Technical Committee and the IEEE Computer Society in 2000, the Best Paper Award from the 2011 IEEE International Electro/Information Technology Conference, the Distinguished University Professorship in 2003, the Faculty of Engineering Deans Special Recognition Award in 2007, and the University of Windsor Award for Excellence in Scholarship, Research and Creative Activity in 2008. He was the IEEE-CAS Representative on the Neural Network Council and the Chair of the IEEE Circuits and Systems Neural Systems Applications Technical Committee in 2000. He has served on the Editorial Board of the *Journal of Circuits, Systems and Computers* as an Associate Editor and a Regional Editor from 1992 to 2012. He has been an Associate Editor for the *Journal of Pattern Recognition* since 1992.