Efficient Hardware Implementation of Large Field-Size Elliptic Curve Cryptographic Processor

CHIOU-YNG LEE, (Senior Member, IEEE), MEDIEN ZEGHID, ANISSA SGHAIER, HASSAN YOUSIF AHMED, AND JIAFENG XIE, (Senior Member, IEEE)

1Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology, Taoyuan City 33306, Taiwan
2Electrical Engineering Department, College of Engineering at Wadi Aldawaser, Prince Sattam Bin Abdulaziz University, Wadi Aldawaser 11991, Saudi Arabia
3Electronics and Micro-Electronics Laboratory, Faculty of Sciences, University of Monastir, Monastir 5000, Tunisia
4Department of Electrical and Computer Engineering, Villanova University, Villanova, PA 19085, USA

Corresponding author: Jiafeng Xie (jiafeng.xie@villanova.edu)

The work of Chiou-Yng Lee was supported by Taiwan MOST under Grant 109-2221-E-262-007-MY2. The work of Jiafeng Xie was supported in part by NSF under Grant SaTC-2020625 and Grant NIST-60NANB20D203.

ABSTRACT Due to rapid development in secured technological devices, the efficient implementation of a large field-size elliptic curve cryptosystem (ECC) is becoming demanding in many critical applications. Therefore, this paper presents a new Montgomery point multiplication (PM) algorithm to optimize and balance the signal flow and resource utilization efficiency. Thereafter, we have presented an efficient ECC processor architecture over $\text{GF}(2^m)$ with $m = 409$ and 571 for the proposed Montgomery PM algorithm. Finally, we have given a detailed comparison and performance analysis (in terms of area-delay product) to show that the proposed cryptographic processor has superior performance as compared to the competing designs. The implementation results after place & route on Xilinx Virtex 7 and Kintex Ultrascale+ are provided. The achieved results reveal that the proposed large field-size ECC processor (and the proposed design strategy) can be extended and applied in many security-demanding applications.

INDEX TERMS Elliptic curve cryptography, FPGA, hardware design, large field-size, low-complexity, point multiplication.

I. INTRODUCTION

Elliptic-curve cryptography [1] was first introduced in the mid-1980s. Compared with traditional RSA (representing Rivest, Shamir, and Adleman), cryptosystems based on elliptic curves have relatively shorter operand lengths while maintaining the same security level (which revolutionized public-key cryptography). Due to the rapid development of security technology, ECC with small field-size has gradually become obsolete. Larger field-size ECCs have thus attracted widespread attention from the research community in recent years.

The most critical operation in ECC is the point multiplication (PM), and generally, a complete PM consists of point addition (PA) and point doubling (PD) operations. The PM thus involves several finite field arithmetic components such as addition, squaring, inversion, and multiplication (multiplication is regarded as the most costly operation, and the inversion can be realized by the multiplicative operation) [2-9]. Besides that, the implementation complexity of the PM is also very much determined by the efficiency of related PM algorithms.

A. EXISTING WORKS

In order to obtain an efficient hardware ECC over $\text{GF}(2^m)$ (binary field ECC is desirable for hardware implementation), many efforts have been carried out. Overall, these works can be categorized into two levels:

(i) System-level, mainly refers to the ECC system implementation (mostly based on the efficient computation of the PM, regarded as the main efforts in the field) [6-12]. Among many algorithms proposed for the realization of PM, the Montgomery algorithm can be seen as the most frequently used one due to its strong attack resistance [3]. It is also noted that many existing reports are mainly focusing on the small field-size ECC, though the National Institute of Standards and Technology (NIST) has recommended five polynomials for ECC application [3] ($\text{GF}(2^{409})$ and $\text{GF}(2^{571})$ are considered as large field-sizes). We have listed several important ECC hardware implementation works below.

In 2008, Chelton et al. [2] designed a high-speed ECC processor. In this paper, a new combined algorithm was
developed to perform PA and PD in an efficient format. Besides that, a subpipelined bit-parallel multiplier was deployed to reduce the latency of PM.

In 2013, Mahdizadeh et al. [4] introduced a highly efficient architecture for ECC PM based on a reorganized and reordered Lopez–Dahab critical-path to achieve maximum architectural and timing improvements. The proposed structures are implemented in parallel, and the critical-path operations are diverted to noncritical paths. The proposed design obtains better efficiency than the previous design.

In 2017, Khan et al. [7] proposed two hardware ECC architectures based on a pipelined full-precision multiplier. To reduce the latency, the authors have proposed a modified Lopez-Dahab Montgomery PM algorithm to avoid data dependency and reduce the required clock cycle numbers. The first single-multiplier-based ECC processor achieves low resource usage, while the second three-multiplier-based ECC processor design obtains the fastest computation time.

In 2018, Li et al. [8] proposed a highly efficient architecture for right-to-left PM algorithm on Koblitz curves to allow parallel computation of Frobenius maps and PAs (to achieve short latency and high frequency). The proposed architecture can perform a point multiplication in 2.50 µs over the five NIST Koblitz curves K-163 at 292 MHz and consume 3,670 slices when implemented on the Virtex-7 device.

In the same year, Imran et al. [9] presented an ECC hardware architecture over GF(2^m) based on BHC (Binary Huff Curves). In this work, a unified and flexible PM hardware architecture based on both ECC and BHC was proposed with the consideration of providing flexibility in the domain of security/reliability. This unified cryptoprocessor obtains better performance than previous ones.

More recently, Imran et al. [10] presented a hardware accelerator for ECC. The authors have proposed an efficient two-stage pipelining architecture with rescheduled PA and PD instructions. It achieves faster computation than the competing ones.

(ii) Component-level, mainly refers to the finite field multiplication and related implementations. In general, the digital-serial finite field multiplier provides a practical compromise in area-time complexities and has attracted the attention of many researchers, i.e., mainstream research work [13-23]. Some of the very recent important works are listed as follows.

In 2006, Kumar et al. [13] introduced different architectural enhancements in the least Significant Digit (LSD) multiplier and their hardware implementations. They proposed the Double Accumulator Multiplier (DAM) architecture and the N-Accumulator Multiplier (NAM) architecture to obtain hardware implementation efficiency.

In 2017, Namin et al. [20] presented two digit-level finite field multipliers in GF(2^m) based on a specific feature of redundant representation in a class of finite fields to minimize hardware resource usage. Theoretical results were then verified by the hardware implementations.

In 2020, José Imana [22] proposed a bit-serial polynomial basis (PB) multipliers architecture over GF(2^m) binary field generated by irreducible trinomials and based on the LFSR (Linear-Feedback Shift Register) technique. It can perform the multiplication in m clock cycles and offers a performance/area trade-off that is very useful in resource-constrained applications.

Besides that, fast algorithms such as Toeplitz Matrix Vector Product (TMVP) have been used to obtain low-complexity implementations, including one of the latest [23], which is superior to other current designs.

From the above discussions, it is desirable that: (i) the proposed highly efficient multipliers over GF(2^m) can be seamlessly integrated with a novel PM algorithm to obtain a perfect ECC implementation; (ii) the employed multipliers and ECC design strategy are ideally suitable for large field-size hardware implementation as it is becoming more demanding recently.

B. MAIN CONTRIBUTIONS

With this point of view, in this paper, we propose an efficient implementation for the large field-size ECC (hardware platform). We derive the proposed ECC through a combination of three coherent interdependent efforts:

(i) An optimized Montgomery PM algorithm is proposed to lay a solid foundation for efficient ECC implementation.

(ii) A new ECC processor is then constructed based on the proposed PM algorithm with the help of several algorithm-architecture co-implementation techniques (i.e., thorough algorithm-to-architecture design details).

(iii) A series of performance comparison and complexity analyses have been carried out to confirm the superior efficiency of the proposed design over the competing ones.

The rest of the paper is organized as follows: The background knowledge is introduced in Section II. Section III derives the proposed PM algorithm. The desired ECC processor is presented in Section IV. The comparison & complexity are provided in Section V. The conclusion is given in Section VI.

II. BACKGROUND INFORMATION

A. ELLIPTIC CURVE CRYPTOGRAPHY

An elliptic curve is the set of points \((x, y)\) over a field \(K\) ([1]) as:

\[ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \]

which can be simplified into the form of: \(y^2 + xy = x^3 + ax^2 + b\) with \(a, b \in K\) over GF(2^m).

Let \(P = (x_1, y_1) \neq O\) (O is the point at infinity) be a point, the inverse of \(P\) is \(-P = (x_1, y_1 + y_1).\) Let \(Q = (x_2, y_2) \neq O\) be a second point with \(Q \neq -P\), the \(P + Q = (x_3, y_3)\) is

\[ x_3 = \frac{y_1 + y_2}{x_1 + x_2} (P \neq Q). \]

Where \(\lambda = \frac{y_1 + y_2}{x_1 + x_2}\). Or

\[ x_3 = \lambda^2 + \lambda + a, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \]

(2)

Where \(\lambda = x_1 + \frac{y_1}{x_1} (P = Q). \)
As seen from (2) and (3), PA and PD both require 1 inversions and 2 multiplications, there is thus a need to find another point representation that can replace the field inversion with field multiplication.

Take a point $E = (x, y)$ to transform $(x, y)$ to affine to projective coordinates $(X, Y, Z)$ with $Z \neq 0$. Thus, for $x = X/Z^a$ and $y = Y/Z^b$, the elliptic curve equation becomes

$$
\frac{Y^2}{Z^b} + \frac{X}{Z^a} \cdot \frac{Y}{Z^b} = \left(\frac{X}{Z^a}\right)^3 + a \cdot \left(\frac{X}{Z^a}\right)^2 + b,
$$

(4)

Where $\alpha$ and $\beta$ should be well chosen that the scalar multiplication requires only multiplication and addition over $GF(2^n)$. Note that here we use the popular Lopez-Dahab (LD) coordinates ($\alpha = 1$ and $\beta = 2$) [5], and (4) can be written as

$$
Y^2 + XYZ = X^3Z + aXZ^2 + bZ^4.
$$

(5)

The choices of appropriate elliptic curves and system coordinates are the first step in the implementation process. Then, the algorithm for the scalar multiplication should be well chosen. Overall, Montgomery’s algorithm is resistant to the side-channel attack because the PA and PD are indistinguishable [24], and we also consider it in this paper.

### B. DIGIT-SERIAL FINITE FIELD MULTIPLIER

First of all, let us consider the fast algorithm of TMVP [25]. Let $V = (V_0, V_1)$ be an $n \times 1$ column vector and the matrix-vector $(T_0, T_1, T_2)$ be used to define an $n \times n$ Toeplitz matrix $T$, where $V_0$ and $V_1$ are two $\frac{n}{2} \times 1$ column vectors, and $T_0$, $T_1$, and $T_2$ are three $\frac{n}{2} \times \frac{n}{2}$ Toeplitz matrices. A TMVP of $C = TV$ in this case is

$$
C = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix} = \begin{bmatrix} T_1 & T_2 \\ T_0 & T_1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix},
$$

(6)

which can be expressed as

$$
\begin{bmatrix} C_0 \\ C_1 \end{bmatrix} = \begin{bmatrix} T_1(V_0 + V_1) + (T_2 + T_1)V_1 \\ T_1(V_0 + V_1) + (T_0 + T_1)V_0 \end{bmatrix}.
$$

(7)

Based on (7), we can recursively generate four components (component matrix point (CMP), component vector point (CVP), point-wise multiply (PWM), and reconstruction (R)) of reduced-size matrices as

$$
\begin{align*}
\text{CMP}(T) &= (T_2 + T_1, T_1, T_0 + T_1), \\
\text{CVP}(V) &= (V_1, V_0 + V_1, V_0), \\
P &= \text{PWM(CMP}(T), \text{CVP}(V)) = (P_0, P_1, P_2), \\
C &= \text{R(P)} = (P_0 + P_1, P_1 + P_2),
\end{align*}
$$

Where $P_0 = (T_2 + T_1)V_1$, $P_1 = T_1(V_0 + V_1)$, $P_2 = (T_0 + T_1)V_0$. Accordingly, Figure 1 shows the subquadratic complexity TMVP-based architecture for (6), which has three stages: the evaluation point generation (EPG) stage, the PWM stage, and the R stage.

The EPG stage performs two block functions of $\text{CMP}(T)$ and $\text{CVP}(V)$, the PWM stage computes $P = \text{PWM(CMP}(T), \text{CVP}(V)) = (P_0, P_1, P_2)$, and the R stage performs the operation $C = \text{R(P)} = (P_0 + P_1, P_1 + P_2)$. Let symbols $S$ and $D$ be “space” and “delay”, respectively, and $S_{D}(n)$ and $S_{D}(n)$ in the case of $n = 2^i (i > 1)$ denote the number of bit-multiplications and the number of bit-additions required for $n \times n$ TMVP multiplication. Meanwhile, let $D_{S}(n)$ and $D_{D}(n)$ denote the number of AND gate delay and the number of XOR gate delay required for TMVP multiplication. In [25], Fan and Hasan have shown that: for 2-way TMVP decomposition, CMP involves $(\begin{array}{c} n \\ 2 \end{array} - 1)$ XOR gates and $T_{\oplus}$ delay (or $T_X$, the delay time of an XOR gate); CVP has $\frac{n}{4}$ XOR gates and $T_{\oplus}$ delay; PWM contains $3S_{D}(\frac{n}{2}) + 3S_{D}(\frac{n}{2})$ space complexity and $D_{S}(\frac{n}{2}) + D_{D}(\frac{n}{2})$ delay; and R unit consists of $n$ XOR gates (delay of $T_{\oplus}$). Accordingly, we have obtained the following recurrences on complexities:

$$
\begin{align*}
S_{D}(n) &= 3S_{D}(\frac{n}{2}), S_{D}(1) = 1 \\
S_{D}(n) &= 3S_{D}(\frac{n}{2}) + 3n - 1, S_{D}(1) = 0 \\
D_{S}(n) &= D_{S}(\frac{n}{2}) + D_{D}(\frac{n}{2}) \\
D_{D}(n) &= D_{D}(\frac{n}{2}) + D_{D}(\frac{n}{2}) = 0
\end{align*}
$$

(8)

To solve the recurrence equations in (8). The time and space complexities of 2-way TMVP decomposition can be expressed as follows ($T_{\ominus}$ and $T_{\oplus}$ are the delay time of AND gate and XOR gate, respectively):

$$
\begin{align*}
S_{D}(n) &= n \log_2^3, \\
S_{D}(n) &= 5.5n \log_2^3 - 6n + 0.5, \\
D(n) &= T_{\ominus} + 2(\log_2 n)T_{\oplus}.
\end{align*}
$$

Now, let $A$ and $B$ be two polynomials in $GF(2^m)$, the polynomial $B$ is grouped by $d$-digit such that $B = B_0 + B_1x^d + \cdots + B_{n-1}x^{d(n-1)}$, where $B_i = \sum_{j=0}^{\left\lfloor \frac{d}{2} \right\rfloor} b_{i,j}x^j$, and $d$ is a power of 2, and $n = \left\lceil \frac{m}{d} \right\rceil$. The product of $A$ and $B$ can be written as (note that we follow the existing notation definition to present the multiplier in [23], which is applicable only in this subsection)

$$
C = AB \mod F(x) = B_0A + B_1Ax^d + \cdots + B_{n-1}A^{d(n-1)} \mod F(x) = B_0A^{(0)} + B_1A^{(1)} + \cdots + B_{n-1}A^{(n-1)},
$$

(9)

Where

$$
A^{(i)} = x^{di}A \mod F(x) = x^{dA^{(i-1)}} \mod F(x).
$$

Following (9), we can firstly use the partial product $AB_i$ to obtain the TMVP formula, and then obtain the digital-serial multiplier with sub-quadratic space complexity.
First, assuming that \( k \) is the number of processing elements (PE), the product in (9) can be rewritten as

\[
C = C_0 + C_1 x^{kd} + \ldots + C_{k-1} x^{d(k-1)} \mod F(x),
\]

(10)

Where

\[
C_i = B_{kdi} A^{(i)} + B_{kdi+1} A^{(i)} x^d + \ldots + B_{kdi+k-1} A^{(i)} x^{d(k-1)} \mod F(x)
\]

\[
A^{(i)} = x^{kd} A^{(i-1)} \mod F(x)
\]

Next, let us define that \( A^{(i)} \) is re-expressed as \( A^{(i)} = T_0^{(i)} + T_1^{(i)} x^d + \ldots + T_n^{(i)} x^{md} \) according to (10). According to the TMVP-based decomposition, the product \( C_i \) is directly transferred as

\[
C_i = \sum_{j=0}^{k-1} R(W_j)x^d,
\]

(11)

Where \((0 \leq k \leq n - 1)\) and

\[
W_j = PWM(CVP(B_{kdi+j}), CMP(A^{(i)}))
\]

\[
= \{W_{j,0}, W_{j,1}, \ldots, W_{j,n}\},
\]

\[
CMP(A^{(i)}) = \{CMP(T_0^{(i)}), CMP(T_1^{(i)}), \ldots, CMP(T_n^{(i)})\},
\]

\[
W_{j,k} = PWM(CVP(B_{kdi+j}), CMP(T_k^{(i)})),
\]

Where we find that \( CMP(A^{(i)}) \) appears in all partial product \( R(W_j) \). In order to reduce the space and time complexities, each partial product \( C_i \) is then split into a two-step computation process:

Step 1: Computing \( P_{Bj} = CVP(B_{kdi+j}) \) for \( 0 \leq j \leq k - 1 \) and \( P_A = CMP(A^{(i)}) \).

Step 2: Computing \( D_{ij} = R(P_{Bj} \odot P_A) \) for \( 0 \leq j \leq k - 1 \) (\( \odot \) is the inner-product operation).

Based on the two-step calculation process, the digital-serial polynomial multiplication over \( GF(2^m) \) is summarized as Algorithm 1. Figure 2 shows the systolic digital-serial multiplication architecture according to Algorithm 1.

**III. PROPOSED LD MONTGOMERY ALGORITHM**

Let two points define as: \( P_1, P_2 \in E(GF(2^m)) \), which are presented in projective coordinates. Meanwhile, define \( P_3, P_4 \) to have: \( P_3 = P_1 + P_2 \) (PA) and \( P_4 = 2 \times P_1 \) (PD). To calculate the mentioned PA and PD, six finite field multiplications, five finite field squaring operations, and four finite field additions are needed [3].

**Proposed Algorithmic Strategy & Details.** We firstly consider that the LD Montgomery algorithm’s computational latency is equivalent to the six field multiplications’ computation time (the field additions and field squaring operations can be simultaneously operated with the multiplications). Besides, we also consider that the performance of ECC is determined by the number of employed multipliers and the digit-size (assume digit-serial multipliers are used to implement the ECC), e.g., the frequency of the processor decreases as the digit-size increases (we can add the number of pipeline stages to improve the maximum operating frequency). Following this strategy, we propose to combine PA and PD to speed up the main computation process, i.e., we propose to only employ two multipliers to achieve low latency implementation, as presented in the proposed Algorithm 2.

**LD Montgomery Algorithm Against Side Channel Attacks.** The scalar multiplication is the most computationally-expensive operation in ECC, which is the primary target of side-channel attacks [26]–[28]. Overall, two aspects of countermeasures are needed to resist side-channel attacks. The first strategy unifies the calculation procedures of elliptic curve PA and PD to make them indistinguishable [29] (from the adversary’s attack). The second technique is adapting the scalar multiplication to make the elliptic curve PA and PD independent of the security bits. The latter aspect includes the double-and-add always method [30] and the Montgomery ladder approach [31]. As clearly stated in the proposed algorithm (Algorithm 2), the Montgomery PM algorithm is highly regular, i.e., there are always two multiplications in each step.
Algorithm 2 Proposed LD Montgomery PM Algorithm (Mul and Sqr denote the multiplication and squaring, respectively)

| Input: | $k=(k_{m-1}, \cdots, k_1, k_0)$ with $k_{m-1} = 1$ |
|--------|--------------------------------------------------|
| $P=(x,y)\in E(F(2^m))$ | Output: $kp=(x_3, y_3)$ |
| Initial Step: | |
| $P(X_1, Z_1) \leftarrow (x, 1)$; $2P=Q(X_2, Z_2) \leftarrow (x^2 + b, x^2)$ | For $i$ from $(m-2)$ to 0 |
| If $k_i = '1'$ then | If $k_i = 1$ then |
| PA: $P(X_3, Z_3) = P(X_1, Z_1) + Q(X_2, Z_2)$; | If $k_i = 0$ then |
| PD: $Q(X_2, Z_2) = 2Q(X_2, Z_2)$; | |
| $S-1$: $Z_1 = \text{Mult}(X_2, Z_1)$; $S-2$: $Z_2 = \text{Mult}(X_1, Z_2)$; $S-3$: $X_3 = \text{Mult}(x, R_3) + \text{Mult}(R_1, R_3)$ | $X_1 = \text{Mult}(X_1, Z_2)$; $X_2 = \text{Mult}(X_2, Z_1)$; $R_1 = \text{Sqr}(Z_2)$; $R_2 = X_2$; |
| $S-1-1$: $Z_2 = \text{Sqr}(Z_2)$; $R_1 = \text{Sqr}(R_1)$; $R_2 = \text{Sqr}(X_2)$; $R_3 = X_1 + Z_1$; | $R_3 = Z_3 = Z_1$; $Z_2 = \text{Mult}(R_2, R_3)$; $R_1 = X_1$; |
| Conversion Step | |
| $(x_3, y_3) = \left(\frac{X_3}{Z_3}, (x + \frac{x_2}{Z_2})((X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)Z_1Z_2)(xZ_1Z_2)^{-1} + y)$ | |

Hence, both PM and PA are performed in every iteration. Therefore, Algorithm 2 is secure in resistant timing attacks and simple power analysis attacks due to the independence between the operation and the value of $k$. Meanwhile, we optimize the modular inversion and modular multiplication algorithms to make the operation time constant to resist timing attacks [32]. Overall, Algorithm 2 makes the proposed ECC processor resistant against simple side-channel attacks (while other types of side-channel attacks are beyond the scope of this paper).

Figure 3 shows the related data flow diagram (based on Algorithm 2). Generally, in projected coordinates, Montgomery PA and PD require six field multiplications, five field squares, and four field addition operations (the delay of the main calculation unit is equal to the delay of six field multiplications). Based on the proposed Algorithm 2, the entire calculation can be decomposed into three steps with only two multipliers at each step (thus reducing the calculation delay and implementation complexity). Please note that we will use the newly released [23] finite field multiplier, which can help us further reduce the involved complexity.

Inversion. Generally, inversion is the most computationally expensive operation in the ECC field [33]. In projective coordinates, an inversion operation is used to convert from projective to affine coordinate. Conventionally, the inversion over $GF(2^m)$ is commonly implemented by the Extended-Euclidean algorithm (EEA) [34] and Itoh–Tsuiji algorithm (ITA) [35]. However, as the conventional EEA-based inversion requires long polynomial division in each iteration, it is thus inefficient for both hardware and software implementations. This problem was partially solved by replacing the degree comparison with a counter [36]. In [37], Yan et al. proposed a modified EEA-based inversion algorithm. In our design, we have used the EEA algorithm of [38].

Algorithm 3 shows the simplified bit-level version of the EEA algorithm, where $m$-bit registers are used to compute five intermediate polynomials $R(x), Y(x), S(x), D(x)$ and $B(x)$. Besides, we have used $R(x)^i$, $Y(x)^i$, $S(x)^i$, $D(x)^i$ and $B(x)^i$ to denote the values of $R(x), Y(x), S(x), D(x)$ and $B(x)$ (after $i$th iteration). Meanwhile, $d_{(i-1)}$ refers to the least significant bit (LSB) of register $D$ (after $(i-1)$th iteration).

Overall, Algorithm 3 has two specified inputs: an element $A$ in $GF(2^m)$ and an irreducible polynomial $F$. The output of the algorithm is multiplicative inverse of $A$, i.e., $B(x)A(x) = 1 \mod F(x)$. Overall, Algorithm 3 uses three steps to deliver the multiplicative inverse result $B$:

- Initial step (Initialization: $i = 0$): coefficients $f_i$ of the irreducible polynomial $F(x)$ and $a_i$ of $A(x)$ are assigned to $R^0$ and $S^0$, respectively. The most significant bit (MSB) of $f(x)$ is always equal ‘1’, and hence there is no need to compute or stored it [36] (which means the coefficients of polynomial $F(x)$ are stored in register of size $m$). Moreover, in this step, $x^{m-1}$ and ‘2’ are assigned to $y^0$ and $D^0$, respectively. Note that the MSBs of $R$ and $S$ are the highest degree terms of $R(x)$ and $S(x)$ ($r_{m-1}, s_{m-1}$), while the LSBs of $Y$ and $B$ are the highest degree terms of $Y(x)$ and $B(x)$ ($y_{m-1}, b_{m-1}$), respectively.
- Updating step: during the whole loop ($1 < i < 2m$) execution time, these five registers are updated by three
Algorithm 3 Bit-Level EEA-Based Inversion Algorithm.

Input: \( A = (a_{m-1}, \ldots, a_1, a_0) \), \( F = (f_{m-1}, \ldots, f_1, f_0) \)
Output: \( B = A^{-1} \mod F \)
Define: \( R = (r_{m-1}, \ldots, r_1, r_0); S = (s_{m-1}, \ldots, s_1, s_0); Y = (y_{m-1}, \ldots, y_1, y_0); D = (d_{m-1}, \ldots, d_1, d_0) \);
\( B = (b_{m-1}, \ldots, b_1, b_0) \);

1. initial step:
\( R^0 = (f_{m-1}, \ldots, f_1, 1); S^0 = A; Y = (0, \ldots, 0, 1); B^0 = 0; D^0 = (2)_2; \) sign\(^0 = 1 \);
2. For (\( i = 0; i \leq 2m; i + + ) \)
// updating control bits
2.1 \( C_1 = i_m^{-1} \);
2.2 \( C_2 = i_{m-1} \otimes \) sign\(^i \);
2.3 sign\(^i = \) not(\( C_1 + \) sign\(^{i-1} \)) + \( d_0^{-1} \otimes \) sign\(^{i-1} \);
// updating registers (\( S, Y, R, B, D \))
2.4 Switch \( C_1 \): 
   a. Case (‘1’):
   i. \( S^i \leq \) SHL(\( S^{i-1}, 1 \)) \( \otimes \) \( R^{i-1} \);
   ii. \( Y^i \leq \) SHL(\( B^{i-1}, 1 \)) \( \otimes \) \( Y^{i-1} \);
   iii. Break;
   b. Case (‘0’):
   i. \( S^i \leq \) SHL(\( S^{i-1}, 1 \));
   ii. \( Y^i \neq Y^{i-1} \);
2.5 Switch \( C_2 \):
   a. Case (‘1’):
   i. \( R^i \leq \) SHL(\( S^{i-1}, 1 \));
   ii. \( B^i \leq Y^{i-1} \);
   iii. Break;
   b. Case (‘0’):
   i. \( R^i \leq R^{i-1} \);
   ii. \( B^i \leq \) SHL(\( B^{i-1}, 1 \));
2.6 Switch sign\(^i \):
   a. Case (‘1’):
   i. \( D^i \leq \) SHL(\( D^{i-1}, 1 \));
   ii. Break;
   b. Case (‘0’):
   i. \( D^i \leq \) SHR(\( D^{i-1}, 1 \));
3. End for;
4. Return \( B \);

IV. PROPOSED ECC PROCESSOR

The proposed ECC processor based on Algorithms 1, 2 and 3 is shown in Figure 4, which consists of the following units:

Main Computation Unit. This unit contains two 7-to-1 MUX, seven registers, one 1-to-7 DeMUX to carry out the necessary operations along with other units in the cryptoprocessor. In particular, these registers and related MUXes/DeMUX coordinate together with the control unit and the arithmetic logic unit for the operations presented in Algorithms 1, 2 and 3.

Arithmetic Logic Unit. This unit focuses on the processing of PA and PD involved in the proposed PM algorithm (through modular multiplication & XORing operations), which constitutes the main data path components in the processor. Based on Algorithm 2, only two multipliers are needed in the proposed design.

Control Unit. This unit generates control signals for all the other units, including the control signals for the data flow in the processor and the movement of data between the Proj vs Aff unit, the main computation unit, and the Aff vs Proj unit. As presented later, we have used a finite state machine (FSM) to generate all the necessary control signals for coordinating all the system-level operations.

Affine-to Projective Coordinates Conversion (Proj vs Aff) Unit. After completing all operations related to scalar multiplication, this unit converts from projective coordinate to affine coordinate. The related result will then be sent to the bus interface unit for further processing. The entire conversion is realized by two multipliers and two inversions.

Affine to Projective Coordinates Conversion (Aff vs Proj) Unit. This unit converts the coordinates of the point \( P(x, y) \) to the point \( P'(X, Y, Z) \). It uses a multiplier and a register to perform the entire conversion, which is achieved by reusing a multiplier and storing the result in each step.

Bus Interface Unit. This unit is responsible for reading input data from the main computing unit and writing output data (adding it to the proposed processor to communicate with the external environment effectively). It is controlled by “start”, “busy”, and “clock” signals, as shown in Figure 4. When “start” signal is set, Affine X, Affine Y and the
key “K” will be received eight-bits by eight-bits (inserting “start” signal after the bus is set that the device is ready to receive data). Once the calculation operation and the conversion step from projective to affine are completed, the result from the main calculation unit will be delivered out eight-bits by eight-bits again.

**Structural Details.** The structural details of the proposed processor are introduced as follows.

1) PA and PD. Based on Algorithm 2 (and Figure 3), the calculation of PA and PD depends on the next key bit \( k_{i+1} \). When \( k_{i+1} = '1' \), the step output will be \( (X_1, Z_1) \) otherwise \( (X_2, Z_2) \) (if \( k_{i+1} = '1' \) and \( X_1 \) & \( Z_1 \) are prepared at the same time, then two multipliers will be passed). When \( k_{i+1} = '0' \), we start with the multiplication between \( X_2 \) and \( Z_1 \) and \( X_1 \) & \( Z_2 \). In step S-1, regardless of \( k_{i+1} \), the square output \( Z_2 \) is stored in the local register \( R_1 \) and the square output \( X_2 \) is stored in the local register \( R_2 \). In step S-2, two multiplications are performed to calculate \( X_2 \) and \( Z_2 \). Moreover, the squaring operation of \( R_3 \) is executed to obtain \( Z_1 \). Before the squaring operation in step S-2, we add \( X_1 \) to \( Z_1 \) in step S-1 to obtain \( Z_1 \). In step S-3, the multiplication between the base point \( x \) and the value in \( R_3 \), and the multiplication between \( R_1 \) and \( R_3 \) are calculated. After that, an addition operation is performed to get a new \( X_1 \).

2) Affine to Projective Coordinates Conversion (Aff vs Proj). The Aff vs Proj unit is controlled by the scalar \( k \), affine coordinates (Affine \( X \) and Affine \( Y \)), irreducible polynomial, and signals of activation (“CLK”, “Start”, and “Reset”). Thus, the Aff vs Proj block converts the affine coordinates to projective ones using two multiplications and one XOR. After conversion, it sends a signal (“Aff_Done”) to the control unit.

3) Control Unit. The control unit is the main component of the proposed ECC processor, which is responsible for all the communications between all components. This unit uses an FSM that the controller synchronizes with the other ECC units, and its details (signal setups) are shown in Figure 5. After receiving the input data \( (k \) and \( P) \) and signals “CLK”, “Reset” = ‘1’, and “Start” = ‘1’, the control unit activates the Aff vs Proj block to convert

\[ P \text{ coordinates from affine to projective coordinates (during conversion, “Clear-Aff” takes ‘1’ & “AffDone” takes ‘0’). If the Aff vs Proj unit completes the conversion (“Clear-Aff” takes ‘0’ and “AffDone” takes ‘1’), the control unit sends signals of activation to the PA and PD blocks (after receiving “AffDone” = ‘1’), respectively, to start calculation (“SynchroPA” = ‘1’, “ResetPA” = ‘1’, “SynchroPD” = ‘1’ and “ResetPD” = ‘1’). After beginning the PA and PD computation, signals “ClearPA” and “ClearPD” take ‘1’ and signals “Done-PA” and “Done-PD” take ‘0’. The PA and PD components are activated firstly by signals (“SynchroPA”, “ResetPA”, “SynchroPD”, and “ResetPD”) sent by the control unit, then they will be activated and disabled by output signals “Clear-PA” and “Clear-PD” to be used repeatedly. When the scalar multiplication is completed, the control unit activates the conversion from projective to affine coordinates. It then sends signals “SynchroProj” = ‘1’ and “Reset-Proj” = ‘1’ to start conversion. The Aff vs Proj block then sets “ClearAff” = ‘1’ and “AffDone” = ‘0’ (after finishing the conversion, “Clear-Aff” takes ‘0’ and “AffDone” takes ‘1’). All the related operands are stored in different register files during the scalar multiplication computation process. Thus, in every step, the control unit loads operands and then stores results for further usage until the scalar multiplication is completed.

4) Projective to Affine Conversion (Proj vs Aff). After the computation of PA and PD \( k \) times, PA and PD send results (“PROJX1”, “PROJZ1”, “PROJX2”, and “PROJZ2”) to the control unit, which activates the Proj vs Aff unit by sending (“PROJX1”, “PROJZ1”, “PROJX2”, and “PROJZ2”), and activating signals (“CLK”, “Reset”, and “SYNCHROPA”), which enables the conversion from projective to affine coordinates (the Proj vs Aff unit then activates its basic operations of multiplication & inversion).

5) Squarer & Inverter. In the proposed ECC processor, squaring is carried out by simply interleaving ‘0’ bits between the original bits [33], as shown in figure 6. Meanwhile, the hardware architecture to execute Algorithm 3 is shown in Figure 7, where the bit-serial inverter uses AND-XOR cells and five \( m \)-bit MUX to update the five registers \( S, Y, R, B, \) and \( D \). The proposed EEA-based inverter has a short critical-path delay and a smaller area as Algorithm 3 has no modular operations.

6) Polynomial Multiplier. In projective coordinates-based ECC implementation, the cryptoprocessor’s overall performance depends on the polynomial multipliers’ performance. We have thus employed the TMVP-based polynomial multiplier of [23] in the proposed ECC processor architecture to obtain low complexity.

7) \( kP \text{ Time (Whole Latency). The multiplier of [23] has a latency of } (2v+2) \text{ cycles, where } v = \sqrt{n} \text{ & } n = \lceil m/d \rceil. \)
While in the proposed design, we have packed the CMP and CVP units of Figure 1 into the product computation unit and the RP cell, thus shortening the design into 2\(v\) cycles. Moreover, according to the data dependence in Figures 2 and 3, the total clock cycles for the proposed ECC processor is:

\[
2v + (m - 1)v + 2m,
\]

Where the scalar multiplication takes \((m - 1)v\) cycles to execute the LD Montgomery PM and \(2m\) cycles to apply inversion (the coordinate conversion consists of two multipliers).

V. COMPLEXITY AND COMPARISON

A. IMPLEMENTATION & COMPLEXITY

We have then implemented the proposed ECC processor on the field-programmable gate array (FPGA) platform based on large field-sizes of \(GF(2^{409})\) and \(GF(2^{571})\). The proposed ECC processor (Figure 4) is coded with VHDL and verified by Modelsim (by using the test vectors provided by the NIST standard [41]). The design is then implemented through Xilinx Vivado 2019.2 (after place & route) on the devices of Virtex 7 (XC7V2000T) and Kintex Ultrascale+ (XCKU15P). The obtained results, namely the maximum frequency (Fmax, MHz), area usage (slices/CLB), \(kP\) time (\(\mu\)s), and area-delay product (ADP = # slices \(\times kP\)) are listed in Table 1. Note that as the static power of the FPGA device takes large portion of the whole power consumption, we thus do not report the power here (this also follows the existing styles in [6], [7], [11], [12], [39], [40]).

B. FPGA COMPARISON

To further evaluate the actual performance of the proposed design, we have also compared the corresponding FPGA implementation results with those of the available reports in the literature, particularly those ones have reported the large field-size [6], [7], [11], [12], [39], [40], as listed in Table 2. To have a fair comparison, we have used the area-delay
product (ADP, i.e., \( \text{ADP} = \# \text{slices} \times kP \) time) as a overall performance metric for all the related designs. We have also calculated the ADP values under different FPGA devices. All the related performance metrics such as area (number of slices), maximum frequency, \( kP \) time, and ADP are all calculated and listed in Table 2. Note that the proposed design of \( d = 16(GF(2^{571})) \) has the most suitable number of PEs in the polynomial multiplier since \( v = \sqrt{n} = \sqrt{m/d} = 6 \), while other choices of \( d \) requires an additional PE to meet the parameter settings, such as \( v = \sqrt{409/16} \approx 5.06 \rightarrow 6 \).

As shown in the table 2, it is clear that the proposed design has superior performance than the existing large field-size ECC implementations, especially those recently released ones [6], [7], [11], [12], [39], [40].

As far as only the latency is concerned, the proposed design over \( GF(2^{409}) \) and \( GF(2^{571}) \) is 32.9\%, and 32.4\% faster than [6], respectively. Besides that, the work presented in [7] over \( GF(2^{571}) \) utilizes 65\% more FPGA slices than this work. As the report based on \( GF(2^{409}) \) is very limited, we thus only compare the results with [6], [12], [39] and find that the proposed ECC processor has significantly less ADP than [6], [12], [39] (even when the results have been adjusted for Virtex-7 and Kintex devices). Considering the maximum field-size recommended by NIST [3], that is, \( GF(2^{571}) \), the proposed design involves much less ADP than the existing ones, especially the latest [11], [39] (the adjusted ADP on the Virtex-7 device is reduced by \( \approx 59.23\% \) and 58.9\%, respectively, according to the results shown in [11], [39]). At the same time, compared with the competing designs of [6], [7], [40] on the Virtex-7 device, the ADP of the proposed processor has 6.02\%, 61.32\%, and 11.09\% smaller ADP, respectively (please note that the overall performance of the design in [7] is actually better than [6], as shown in [7]).

The superior performance of the proposed ECC processor benefited from: (i) proper arrangement of the signal flow & resource usage brought by the proposed LD Montgomery algorithm; (ii) proposed algorithm-architecture co-implementation techniques for the ECC processor. Further efforts can be made to optimize the finite field multiplier to obtain higher efficiency.

**VI. CONCLUSION**

In this paper, we propose three new efforts to obtain an effective hardware implementation of a large field-size ECC processor: (i) We firstly propose a novel LD Montgomery PM algorithm to arrange proper signal flow for ECC; (ii) Then, we construct a new ECC processor based on a series of algorithm-architecture co-implementation techniques; (iii) Lastly, we carried out detailed implementation-based complexity analysis and comparison to prove the effectiveness of the proposed ECC processor. The proposed ECC processor is highly efficient, and it can be extended for deploying in many critical applications. Future work may focus on the developing of more efficient finite field multipliers and related cryptographic processors.

**REFERENCES**

[1] V. Miller, “Use of elliptic curves in cryptography,” in Advances in Cryptology—CRYPTO (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 1986, pp. 417–426.

[2] W. N. Chelton and M. Benaissa, “Fast elliptic curve cryptography on FPGA,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* vol. 16, no. 2, pp. 198–205, Feb. 2008.

[3] Standards for Efficient Cryptography 2 (SECC2. Recommended Elliptic Curve Domain Parameters, Certicom Research, 2010.

[4] H. Mahdizadeh and M. Masoumi, “Novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over \( GF(2^{163}) \),” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* vol. 21, no. 12, pp. 2330–2333, Dec. 2013.

[5] J. Lopez and R. Dahab, “Improved algorithms for elliptic curve arithmetic in \( GF(2^n) \),” in Selected Areas in Cryptography (SAC) (Lecture Notes in Computer Science), vol. 1556, S. Tavares and H. Meijer, Eds. Berlin, Germany: Springer, 1998, pp. 201–212.

[6] Z. Khan and M. Benaissa, “Throughput/area-efficient ECC processor using Montgomery point multiplication on FPGA,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 11, pp. 1078–1082, Nov. 2015.

[7] Z. Khan and M. Benaissa, “High-speed and low-latency ECC processor implementation over \( GF(2^{254}) \) on FPGA,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 165–176, Jan. 2017.

---

**TABLE 1.** FPGA Implementation Performance for the Proposed ECC Processor (after Place and Route) in Virtex-7 & Kintex Ultrascale+ Devices.

| Design | FPGA | MS $^1$ | Slices | Fmax (MHz) | kP | ADP |
|--------|------|---------|--------|------------|----|-----|
| Proposed ECC processor over \( GF(2^{571}) \) (Virtex-7) |
| \( d = 8 \) | 57,678 | 23,351 | 17,324 | 147 | 38.89 | 673 |
| \( d = 16 \) | 58,788 | 23,141 | 16,617 | 112 | 40.83 | 678 |

| Proposed ECC processor over \( GF(2^{571}) \) (Kintex Ultrascale+) |
| \( d = 8 \) | 39,145 | 17,726 | 7,574 | 260 | 12.0 | 92 |
| \( d = 16 \) | 45,111 | 16,774 | 8,078 | 211 | 13.59 | 109 |

| Proposed ECC processor over \( GF(2^{571}) \) (Kintex Ultrascale+) |
| \( d = 8 \) | 57,567 | 23,379 | 10,399 | 161 | 35.51 | 369 |
| \( d = 16 \) | 58,852 | 23,139 | 10,922 | 142 | 32.21 | 351 |

*ADP: \( \times 10^3 \). *#CLBs in Kintex Ultrascale+ device.

**TABLE 2.** Comparison of the Proposed and the Existing ECC Processors (after Place and Route).

| Design | FPGA | MS $^1$ | Slices | Fmax (MHz) | kP | ADP |
|--------|------|---------|--------|------------|----|-----|
| ECC over \( GF(2^{571}) \) |
| [39] | Virtex-6 | 286 | 29,000 | 135 | 41.36 | 1,199 |
| [6] | Virtex-7 | 143 | 6,888 | 316 | 32.72 | 226 |
| [12] | Virtex-5 | 409 | 6,254 | 164 | 142.6 | 892 |
| Ours | Virtex-7 | 409 | 11,129 | 168 | 21.95 | 244 |
| Ours | Kintex Ultra+ | 409 | 7,374 | 260 | 12.8 | 92 |

| ECC over \( GF(2^{571}) \) |
| [39] | Virtex-6 | 286 | 29,000 | 135 | 56.50 | 1,658 |
| [40] | Virtex-5 | 571 | 18,645 | 143 | 40.6 | 757 |
| [12] | Virtex-7 | 571 | 8,707 | 128 | 325.2 | 3,069 |
| [6] | Virtex-7 | 571 | 12,965 | 250 | 57.6 | 747 |
| [7] | Virtex-7 | 571 | 50,336 | 111 | 43.05 | 1,815 |
| [11] | Virtex-7 | 571 | 7,876 | 227 | 218.1 | 1,717* |
| Ours | Virtex-7 | 571 | 17,324 | 147 | 38.89 | 673 |
| Ours | Kintex Ultra+ | 571 | 10,922 | 142 | 32.21 | 351 |

*can be adjusted to ADP value on the Virtex-7 device as \( \approx 1,346 \), according to the results presented in [11].

1. MS: multiplier size (involved finite field multiplier). 2. U+: Ultrascale+.
[8] L. Li and S. Li, “High-performance pipelined architecture of point multiplication on Koblitz curves,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 65, no. 11, pp. 1723–1727, Nov. 2018.

[9] M. Imran, M. Rashid, A. R. Jafari, and M. Najam-ul Islam, “Acryp-Proc: Flexible asymmetric crypto processor for point multiplication,” IEEE Access, vol. 6, pp. 22778–22793, 2018.

[10] M. Imran, S. Pagliarini, and M. Rashid, “An area aware accelerator for elliptic curve point multiplication,” in Proc. 27th IEEE Int. Conf. Electron., Circuits Syst. (ICECS), Glasgow, U.K., Nov. 2020, pp. 1–4.

[11] S. Harb and M. Jarrah, “FPGA implementation of the ECC over GF(2^m),” for small embedded applications,” ACM Trans. Embedded Comput. Syst., vol. 18, no. 2, pp. 1–19, Mar. 2019.

[12] G. D. Sutter, J. Deschamps, and J. L. Imaña, “Efficient elliptic curve point multiplication using digit-serial binary field operations,” IEEE Trans. Ind. Electron., vol. 60, no. 1, pp. 217–225, Jan. 2013.

[13] S. Kumar, T. Wollinger, and C. Paar, “Optimum digit serial GF(2^m) multipliers for curve-based cryptography,” IEEE Trans. Comput., vol. 55, no. 10, pp. 1306–1311, Oct. 2006.

[14] C.-Y. Lee and J. Xie, “Digit-serial versatile multiplier based on a novel block recombination of the modified overlap-free Karatsuba algorithm,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 6, pp. 203–214, Jan. 2019.

[15] 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.

[16] C.-Y. Lee, C.-C. Fan, J. Xie, and S.-M. Yuan, “Efficient implementation of Karatsuba algorithm based three-opendar multiplication over binary extension field,” IEEE Access, vol. 6, pp. 38234–38242, 2018.

[17] P. Meher and X. Lou, “Low-latency, low-area, and scalable systolic-like modular multipliers for GF(2^m) based on irreducible all-one polynomials,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 64, no. 2, pp. 399–408, Feb. 2017.

[18] J. Xie, P. K. Meher, X. Zhou, and C.-Y. Lee, “Low-register-complexity systolic digit-serial multiplier over based on trinomials,” IEEE Trans. Multi-Scale Comput. Syst., vol. 4, no. 4, pp. 773–783, Oct. 2018.

[19] M. Morales-Sandoval, C. Feregrino-Tribe, P. Kitsos, and R. Cumplido, “Area/performance trade-off analysis of an FPGA digit-serial GF(2^m) Montgomery multiplier based on LFSR,” Comput. Electr. Eng., vol. 39, no. 2, pp. 542–549, 2013.

[20] P. Namin, R. Muscedere, and M. Ahmadi, “Digit-level serial-in parallel multipliers for extended binary fields,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1632–1643, May 2017.

[21] J. Xie, C. Lee, P. Meher, and Z.-H. Mao, “Novel bit-parallel and digit-serial systolic finite field multipliers over GF(2^m) based on reordered normal basis,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 7, no. 9, pp. 2119–2130, Sep. 2019.

[22] J. L. Imaña, “LFSR-based bit-serial GF(2^m) multipliers using irreducible trinomials,” IEEE Trans. Comput., vol. 70, no. 1, pp. 156–162, Jan. 2021.

[23] L. Fan, C.-Y. Lee, and R. Zeghidi, “Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix-vector product approach,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 7, pp. 1614–1622, Mar. 2019.

[24] J. López and R. Dahab, “Fast multiplication on elliptic curves over GF(2^m) without precomputation,” in Proc. CHES, vol. 1717, 1999, pp. 316–327.

[25] H. Fan and M. A. Hasan, “A new approach to subquadratic space complexity parallel multipliers for extended binary fields,” IEEE Trans. Comput., vol. 56, no. 2, pp. 224–233, Feb. 2007.

[26] J.-L. Danger, S. Guille, P. Hoogvorst, C. Murdica, and D. Naccache, “A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards,” J. Cryptographic Eng., vol. 3, no. 4, pp. 241–265, Nov. 2013.

[27] K. Liao, X. Cui, N. Liao, T. Wang, D. Yu, and X. Cui, “High-performance noninvasive side-channel attack resistant ECC coprocessor for GF(2^m),” IEEE Trans. Ind. Electron., vol. 64, no. 1, pp. 727–738, Jan. 2017.

[28] Z. Liu, D. Liu, and X. Zou, “An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor,” IEEE Trans. Ind. Electron., vol. 64, no. 3, pp. 2353–2362, Mar. 2017.

[29] D. Stebila and N. Thériault, “Unified point addition formulae and side-channel attacks,” in Cryptographic Hardware and Embedded Systems (CHES), vol. 4249. Yokohama, Japan: Springer, 2006, pp. 354–368.

[30] J. Coron, “Resistance against differential power analysis for elliptic curve cryptosystems,” in Cryptographic Hardware and Embedded Systems (CHES), vol. 1717. Worcester, MA, USA: Springer, 1999, pp. 292–302.

[31] M. Joyce and S. Yen, “The Montgomery powering ladder,” in Cryptographic Hardware and Embedded Systems (CHES), vol. 2523. Redwood Shores, CA, USA: Springer, 2003, pp. 291–302.

[32] J. Fan, L. Batina, and I. Verbauwhede, “Design and design methods for unified multiplier and inverter and its application for HECC,” Integration, vol. 44, no. 4, pp. 280–289, 2011.

[33] S. Liu, J. Ju, X. Cai, Z. Jia, and Z. Zhang, “High performance FPGA implementation of elliptic curve cryptography over binary fields,” in Proc. 13th Int. Conf. Trust, Secur. Privac. Comput. Commun., Beijing, China, Sep. 2014, pp. 148–155.

[34] D. Knuth, The Art of Computer Programming. Seminumerical Algorithms, vol. 2. Boston, MA, USA: Addison-Wesley, 1981.

[35] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases,” Inf. Comput., vol. 78, no. 3, pp. 171–177, Sep. 1988.

[36] R. P. Brent and H. T. Kung, “Systolic VLSI arrays for polynomial GCD computation,” IEEE Trans. Comput., vol. 33, no. 8, pp. 731–736, Aug. 1984.

[37] Z. Yan, D. V. Sarwate, and Z. Liu, “High-speed systolic architectures for finite field inversion,” Integration, vol. 38, no. 3, pp. 383–398, 2005.

[38] A. Ibrahim, T. F. Al-Somani, and F. Gebali, “New systolic array architecture for finite field inversion,” Can. J. Electr. Comput. Eng., vol. 40, no. 1, pp. 23–30, winter 2017.

[39] X. Zhao, B. Li, L. Zhang, Y. Wang, Y. Zhang, and R. Chen, “FPGA implementation of high-efficiency ECC point multiplication circuit,” Electron. Lett., vol. 10, no. 11, p. 1252, 2021.

[40] L. Li and S. Li, “High-performance pipelined architecture of elliptic curve scalar multiplication over GF(2^m),” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 4, pp. 1223–1232, Apr. 2016.

[41] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, “A statistical test suite for random and pseudo-random number generator for cryptographic applications,” NIST Pubs, Gaithersburg, MD, USA, Tech. Rep. 800-22 Rev 1a, 2010.
ANISSA Sghaier received the B.Eng. degree in real time computer sciences engineering from Sousse University, Tunisia, in 2010, and the M.Sc degree in electronics and micro-electronics and Ph.D. degree in information technologies from the University of Monastir, in 2012 and 2016, respectively. She is currently a Research Member of the Laboratory of Electronics and Micro-electronics, Faculty of Science of Monastir. Her research interests include information security, software/hardware implementation of cryptographic cryptosystems, and image and video coding.

HASSAN YOUSSIF AHMED received the B.Eng. degree in computer engineering (network systems) and the M.Sc. degree in computer science and information from Gezira University, Sudan, in 2002 and 2007, respectively, and the Ph.D. degree in electrical and electronic engineering from University Tecknolgi PETRONAS, Malaysia, in 2010. He is currently an Associate Professor with the Electrical Engineering Department, College of Engineering, Prince Sattam Bin AbdulAziz University. He is also the Dean Assistant of Quality and Development. His research interests include computer networks, wireless communications networks, optical communications, and cryptography systems.

JIAFENG (HARVEST) XIE (Senior Member, IEEE) received the M.E. degree from Central South University, in 2010, and the Ph.D. degree from the University of Pittsburgh, in 2014. He is currently an Assistant Professor with the Department of Electrical & Computer Engineering, Villanova University, Villanova, PA, USA. His research interests include cryptographic engineering, hardware security, post-quantum cryptography, and VLSI implementation of neural network systems.

Dr. Xie has served as a technical committee member for many reputed conferences, such as HOST, ICCAD, and DAC. He is also currently serving as an Associate Editor for Microelectronics Journal and IEEE Access. He was serving as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS. He received the IEEE Access Outstanding Associate Editor for the year of 2019. He also received the Best Paper Award from HOST’19.

* * *