Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8883614, 7 pages
https://doi.org/10.1155/2021/8883614

*Research Article*

# A Hardware-Efficient Elliptic Curve Cryptographic Architecture over GF (p)

**Chao Cui, Yun Zhao, Yong Xiao, Weibin Lin, and Di Xu**

*Electric Power Research Institute of CSG, Guangzhou, Guangdong 510663, China*

Correspondence should be addressed to Chao Cui; ncepucc@126.com

Received 10 September 2020; Revised 15 January 2021; Accepted 5 May 2021; Published 19 May 2021

Academic Editor: Hussein Abulkasim



This paper proposes a hardware-efficient elliptic curve cryptography (ECC) architecture over GF(p), which uses adders to achieve scalar multiplication (SM) through hardware-reuse method. In terms of algorithm, the improvement of the interleaved modular multiplication (IMM) algorithm and the binary modular inverse (BMI) algorithm needs two adders. In addition to the adder, the data register is another optimize target. The design compiler is synthesized on 0.13 $\mu$m CMOS ASIC platform. The time range of performing scalar multiplication over 160, 192, 224, and 256 field orders under 150 MHz frequency is 1.99–3.17 ms. Moreover, the gate area required for different field orders in this design is in the range of 35.65k–59.14k, with 50%–91% hardware resource less than other processors.

## 1. Introduction

Due to the rapid development of technology, Internet of Things- (IoT-) related devices have become popular. Most importantly, the safety must be guaranteed. In addition to the IoT devices, the safety of road networks also needs to be paid great attention [1]. Miller [2] and Koblitz [3] put forward the concept of elliptic curve cryptography (ECC), which is a kind of asymmetrical cryptosystem put forward by Miller [2] and Koblitz [3] in 1986, which has higher security than other methods like RSA encryption algorithm. Several international organizations have adopted ECC, including NIST [4], ANSI [5] and IEEE [6].

For ECC, there have been a large number of hardware architectures [7–17]. Among them, there are two methods for the realization of modular multiplication (MM), namely, the multiplier and the adder. The multiplier-based architecture includes the design based on specific prime field and the design based on Montgomery multiplication algorithm [7]. The adder-based architecture includes the design based on interleaved multiplication algorithm [9]. The processor [13] uses a design with Montgomery MM algorithm and r-bit $*$ r-bit multiplier. The processors [8, 14] use a design with n-bit $*$ n-bit multiplier. MM includes multiplication and fast reduction operation over a specific prime field. It should be noted that the multiplier-based architecture requires a lot of hardware.

In ECC, modular inversion (MI) is also a kind of cumbersome operation. Among them, binary modular inversion algorithms are usually used in hardware-efficient architectures. The MM and MI units of processor [11] are based on the adder, and the two units are independent in adder. Processors [18, 19] adopt a radix-4 booth encoding IMM algorithm. Processor [20] implements MM through a radix-2 MM algorithm and avoids MI through projective coordinates.

Traditional cryptographic algorithm software has the disadvantages of high power consumption and time delay, which can be solved by hardware implementation. This article attempts to provide security assurance with low

power consumption for IoT devices through hardware implementation. The following are the main contributions of this article.

(1) A hardware-efficient architecture based on add units is proposed to achieve as little hardware consumption as possible

(2) Through the modification of IMM algorithm and BMI algorithm with 2 full-word adders and four data registers, MM and MI can be realized

(3) Registers are optimized to minimize hardware consumption, in which four full-word register units for MM, MS, MA, and MI and eight full-word register units for SM operation

The structure of this article is divided into four parts. First, the Mathematical Background section elaborates on EC operation and SM operation. Second, the Scalar Multiplication Architecture section introduces the hardware-efficient architecture over GF(p). Third, the Implementation and Result section shows the results and then conducts comparative analysis. Fourth, the Conclusion section is a summary.

## 2. Mathematical Background

*2.1. Elliptic Curve over GF(p).* An introduction on EC over GF(p) is conducted. When the $p$ value of nonsupersingular elliptic curve $E$ on GF(p) is greater than 3, the following formula can be used:

$$y^2 = x^3 + ax + b, \tag{1}$$

where $a$, $b$, $x$, and $y$ are elements of GF(p), and $4a^3 + 27b^2 \neq 0 \bmod p$. See [21, 22] for more information on elliptic curve cryptographic primitives.

Formulas (2) and (3) show the point adding (PA) operation and point doubling (PD) operation. With elliptic curve point $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, the computing formula for PA is $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$, and the computing formula for PD is $P_3(x_3, y_3) = 2P_1(x_1, y_1)$.

$$\begin{cases} x_3 = \lambda^2 - x_1 + x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}, \tag{2}$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P_1 \neq P_2, \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } P_1 = P_2. \end{cases} \tag{3}$$

*2.2. Elliptic Curve Scalar Multiplication.* In ECC, SM is the basic operation. As for PM operation, integer $k$ and point P on the elliptic curve are input and then performed as a sequence of PA and PD operations given in Algorithm 1. In Step 1, the point $Q$ is initialized as a point at infinity. Step 2 is to iterate $n-1$ times, where each iteration has the PD operation. $k_i = 1$ indicates that there is a PA operation.

## 3. Scalar Multiplication Architecture

This part describes the bottom-up algorithm optimization on GF(p), which achieves maximum reuse by adder unit. The SM operation is implemented by using two full-word adder units. The optimization of MM and MI operations is conducive to the reduction of power consumption and the improvement of SM operation's performance.

*3.1. Modular Addition/Subtraction.* MA and MS operations are implemented based on Algorithm 2. In ASIC, the addition or subtraction operations can be implemented using nearly equal hardware, namely, adder units. Since MA and MS operations require a clock cycle, there is a need for 2 full-word adders. In addition, here, the adder unit is the minimum unit, and $C0_n$ and $C1_n$ are the most significant bits (MSB).

*3.2. Modular Multiplication.* MM is an indispensable operation in SM operation architecture. In this study, the interleaved modular multiplication algorithm is selected. The standard interleaved modulo multiplication in [16] (Algorithm 2) has certain shortcomings. Since steps 5, 6, and 7 carry out addition operations with carry propagation and steps 6 and 7 check all lengths of the operands, there is a large latency. In response to this problem, the improved algorithm in [16] (Algorithm 3) performs addition operations with carry-save adders in the loop. Moreover, the modified algorithm in [16] (Algorithm 4) reduces the area and time by lookup-table method. In [10], a new interleaved modular multiplication algorithm is proposed, which uses only two adder units. The specific steps are shown in Algorithm 5 as follows. In step 1, the variable $R$ is initialized to zero. In step 2.1, the $R * 2$ can be realized by shifting operation. In step 2.2, the $X_i * Y$ can be implemented by a multiplexer. Step 2.3 and step 2.4 require an adder unit, respectively. Therefore, if each iteration is completed within one clock cycle, then a total of two adder units are required. After the iteration of step 2, the result is limited to $[0, 2p-1]$. Therefore, it is necessary to go to step 3 to limit $R$ to $[0, p-1]$.

*3.3. Modular Inversion.* In addition to the MM operation, the modular inversion (MI) operation also plays an extremely important role in the SM operation architecture. In MI operation, the same two adder units are reused to reduce hardware consumption. This paper adopts the binary modular inversion algorithm proposed in [10]. Algorithm 3 can calculate MM and MI operations in the same clock cycle. If the input $a = 1$, it is an MI operation, and $y = 1/x \bmod p$. In step 1, the variables $u$, $v$, $r$, $s$ are initialized. In step 2 and step 3, the /2 operations can be realized by right shifting one bit. With a positive or negative odd $r$, $R/2 \bmod p = (r + p) \gg 1$, that is, it can be computed by adding $r$ to $p$ and then shifting right. The same is true in other situations. The above operations require one adder unit. In step 2 or step 3, the comparison between $u$ and $v$ in step 4 is calculated in advance, which requires two adder

```
Input: an integer k and a point P on elliptic curve
Output: kP
(1) Q = ∞;
(2) for (i = n − 1; i ≥ 0; i + + ) {
    (2.1) Q = 2Q;
    (2.2) if k_i = 1, {Q = Q + P; }
          }
(3) return Q
```

Algorithm 1: Elliptic curve scalar multiplication.

```
Input: p, A, B ∈ [0, p − 1]
Output: R = (A + B) mod p
(1) C0 = A + B
(2) C1 = C0 − p
(3) if C1n = 1 {R = C0}
(4) else {R = C1}
(5) return R
Input: p, A, B ∈ [0, p − 1]
Output: R = (A − B) mod p
(1) C0 = A − B
(2) C1=C0 + p
(3) if C0 n = 1 {R = C1}
(4) else {R = C0}
(5) return R
```

Algorithm 2: Modular addition and subtraction in GF(p).

```
Input: p, x, a ∈ [1, p − 1]
Output: y, satisfying xy = a mod p
Step 1: u = p; v = x; r = 0; s = a;
Step 2: if (u is even) {
   u = u/2;
   if (r is odd) r = (r + p)/2;
   else if (r is negative) r = (r + 2p)/2;
   else r = r/2;
   }
Step 3: if (v is even) {
   v = v/2;
   if (s is odd) s = (s + p)/2;
   else if (s is negative) s = (s + 2p)/2;
   elses = s/2;
   }
Step 4: if (u and v are odd) {
   if(u > v) r = r − s; u = u − v;
   else s = s − r; v = v − u;
   }
Step 5: if (u = 1) {
      if (r < 0) {return r = r + p; }
      else { return r.}
   }
   else if (v = 1) {
      if (s < 0) {return s = s + p; }
      else {return s.}
   }
   else {go to step2.}
```

Algorithm 3: Binary modular inversion (IBMI) algorithm.

Input: P1($x1$, $y1$), P2($x2$, $y2$),
Output: P3($x3$, $x3$) = $P1 + P2$
(1) $t2 = y1 - y2$
(2) $t1 = x1 - x2$
(3) $t1 = t2/t1$
(4) $t2 = t1 * t1$
(5) $t2 = t2 - x1$
(6) $t2 = t2 - x2$
(7) $t2 = x2 - t2$, $x1 = t2$
(8) $t2 = t1 * t2$
(9) $y1 = t2 - y2$
(10) return $x3 = x1, y3 = y1$
Input: P1($x1$, $y1$) = P2($x2$, $y2$)
Output: P3($x3$, $x3$) = $P1 + P2$
(1) $t2 = x1 * x1$
(2) $t1 = t2 + t2$
(3) $t1 = t2 + t1$
(4) $t2 = t1 + a$
(5) $t1 = y1 + y1$
(6) $t1 = t2/t1$
(7) $t2 = t1 * t1$
(8) $t2 = t2 - x1$
(9) $t2 = t2 - x2$
(10) $t2 = x2 - t2$, $x1 = t2$,
(11) $t2 = t1 + t2$
(12) $y1 = t2 - y2$
(13) return $x3 = x1$, $y3 = y1$

Algorithm 4: Point addition and point doubling.

Input: $p, X, Y \in [0, p - 1]$
Output: $R = X * Y \bmod p$
(1) $R = 0$;
(2) for{($i = n - 1$; i $\geq 0$; i + +)
(2.1) $R = R * 2$;
(2.2) $I = Xi * Y$;
(2.3) $R = R + I$;
(2.4) $R = R - R[n + 1: n] * p$;
}
(3) if $R \geq p$ {$R = R - p$; }
(4) return $R$

Algorithm 5: Interleaved modular multiplication algorithm.

units. In step 4, $(r - s, u - v)$ or $(s - r, v - u)$ is calculated, which requires two adder units. Step 5 requires a total of two adder units, one of which is used to determine whether $r$ or $s$ is less than 0, and the other is used for $r + p$ or $s + p$. Therefore, if each step is completed in one clock cycle, two adder units are required.

*3.4. Point Addition and Point Doubling.* Algorithm 4 provides PA and PD operations. Since modular operations (MA, MS, MM, and MI) share the same two adder units, only one modular operation is computed at a time. A total of eight registers are required, of which six are used for PA and PD operations of $t_1$, $t_2$, $x_1$, $x_2$, $y_1$, and $y_2$, and two for integer $k$ and prime $p$.

*3.5. Scalar Multiplier Architecture.* In this part, Figure 1 shows the scalar multiplication architecture of SM on GF(p), which achieves the modular operations of MM, MS, MA, and MI as well as the point operations of SM, PA, and PD. Among them, point controller block is the main state

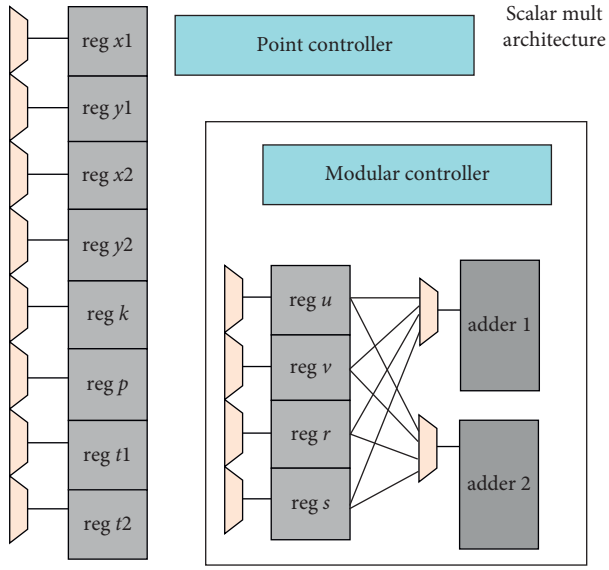


Figure 1: Scalar multiplication architecture.

machine that realizes the point operation, and modular controller block is the state machine that realizes the modular operation.

## 4. Implementation and Result

The ECC architecture described in this part is designed using Verilog-HDL language and adopt Design Compiler to synthesize it using SMIC 130-nm CMOS standard cell library. In addition, the experimental circuit area is evaluated by the 2-way NAND gate.

The source of the experimental simulation parameters is the FIPS 186-2 standard [8]. Figure 2 lists the main parameters for one 256 bit elliptic curve on the prime field GF(p) and the other bit elliptic curve can be found in the FIPS 186-2 standard. The coordinates of base point $G$ on elliptic curve are $Gx$ and $Gy$.

| Parameter | Value |
|---|---|
| $a$ | FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC |
| $b$ | 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53130F6 3BCE3C3E 27D2604B |
| $p$ | FFFFFFFF 000000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF |
| $Gx$ | 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 |
| $Gy$ | 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5 |

Figure 2: Curve parameter of our experiment.

There is a need for a total of two adders and twelve data registers in the proposed architecture. According to Table 1, the required registers and adders consumed 42% of the hardware. Among them, the twelve registers are used for data storage. With the increase of field order, the adder's resource consumption percentage increases from 13.72% to 15.54%.

In Table 2, the results of the implementation and comparison of the proposed architecture are shown. By testing 100 times, the SM operation requires an average of 186, 268, 364, and 475 clock cycles on 160, 192, 224, and 256 prime fields, respectively. The proposed architecture takes 1.24, 1.78, 2.42, and 3.16 ms with the 35.65 k, 43.25 k, 49.41 k, and 59.14 k gate area for one SM operations over 160, 192, 224, and 256 prime fields, respectively.

The authors of [10, 11] use IMM algorithm and BIA to realize the inversion and multiplier units. Among them, the processor in [10] and the processor we proposed use the same method, that is, use the same unit to implement MM and MI operations. But in contrast, the proposed design has higher performance on the prime fields of 160/192/224/256 bits, which is 1.28~1.29 times faster than that of [10]. Under 160 bit prime field, the processor in [10] takes 35.43k gate area and 1.60 ms to perform an SM operation. In the area-time product (AT) parameter, the AT value of the processor we designed is relatively low, indicating that there is a better balance between hardware consumption and performance.

The processor in [11] uses two adder-based inversion units and two adder-based multiplier units, and our processor uses one combined unit. In contrast, the proposed processor has the advantage of low hardware consumption. In addition, our design saves 64.81%, 64.87%, 65.66%, and 64.69% area over the 160/192/224/256 bit prime fields than the design in [11]. Taking the 160 bit prime field as an example, the processor in [11] takes 101.3 k gate area and

Table 1: Hardware consumption of register and adder.

| Field order | Total area | Area | | | Percent | | |
|---|---|---|---|---|---|---|---|
| | | Register | Adder | Register/adder | Register | Adder | Register/adder |
| 160 | 35.65 | 9.95 | 4.89 | 14.84 | 27.91 | 13.72 | 41.63 |
| 192 | 43.25 | 11.83 | 6.19 | 18.02 | 27.35 | 14.31 | 41.66 |
| 224 | 49.41 | 13.69 | 7.02 | 20.71 | 27.71 | 14.21 | 41.91 |
| 256 | 59.14 | 15.87 | 9.19 | 25.06 | 26.83 | 15.54 | 42.37 |

Table 2: ECC hardware performance comparison.

| Design | Technology | Field order | Area (kgate) | Frequency (MHz) | Clock cycles (k) | SM (ms) | At |
|---|---|---|---|---|---|---|---|
| This work | 0.13 $\mu$m CMOS | 160 | 35.65 | 150 | 186 | 1.24 | 44 |
| | | 192 | 43.25 | 150 | 268 | 1.78 | 77 |
| | | 224 | 49.41 | 150 | 364 | 2.42 | 120 |
| | | 256 | 59.14 | 150 | 475 | 3.16 | 187 |
| 10 | 0.13 $\mu$m CMOS | 160 | 35.43 | 150 | 239 | 1.60 | 57 |
| | | 192 | 43.37 | 150 | 342 | 2.28 | 99 |
| | | 224 | 50.38 | 150 | 468 | 3.12 | 157 |
| | | 256 | 57.05 | 150 | 610 | 4.07 | 232 |
| 11 | 0.13 $\mu$m CMOS | 160 | 101.3 | 150 | 129.3 | 0.87 | 88 |
| | | 192 | 123.1 | 138 | – | 1.36 | 167 |
| | | 224 | 143.9 | 130 | – | 1.95 | 281 |
| | | 256 | 167.5 | 110 | – | 3.01 | 504 |
| 13 | 0.13 $\mu$m CMOS | 160 | 117.5 | 137.7 | 153 | 1.21 | 142 |
| | | 192 | 118.02 | 137.7 | 184 | 1.44 | 170 |
| | | 224 | 120.26 | 137.7 | 297 | 2.34 | 281 |
| | | 256 | 120.26 | 137.7 | 340 | 2.68 | 322 |
| 14 | 0.13 $\mu$m CMOS | 256 | 659 | 163.7 | 3.3 | 0.02 | 13 |
| 15 | 0.13 $\mu$m CMOS | 256 | 122 | 556 | 562 | 1.01 | 123 |

0.87 ms. Although it has higher performance, the design we propose chooses a lower AT value in order to balance hardware consumption and performance. In summary, the proposed processor has the advantages of low hardware consumption and high hardware efficiency.

The processor in [13] uses a word-based Montgomery multiplier and dynamic redundant binary converter, which can improve the performance of SM. Compared with the design in [13], our design can save 69.66%, 63.35%, 58.93%, and 50.84% area over the 160/192/224/256 bit prime fields.

The processor in [14] causes large power consumption, which is not suitable for IoT devices. More specifically, a full-size 256 bit × 256 bit multiplier requires a large hardware consumption, namely, 659 k gate. In contrast, the proposed design can save 91.03% of the area. The processor in [15] uses a systolic arithmetic unit in high frequency of 556 MHz. Based on the 256 bit prime fields, our design can save 51.52% of the area.

Compared with the abovementioned processors in [10, 11, 13, 14] and [15], our proposed processor has the least hardware consumption.

## 5. Conclusion

By constructing a bottom-up optimization for all operations of algorithm-level scalar multiplication on the basis of two full-word adders, a hardware-efficient elliptic curve processor over GF(p) is proposed. Through the improvement of IMM algorithm and BMI algorithm, they become suitable for two adder units. Moreover, the registers are also optimized. A total of 12 full-word register units are used to store data. Synthesized on 0.13 $\mu$m ASIC platform, the processor's hardware consumption can be controlled within the range of 35.65 k~59.14 k, which is far lower than most processors.

## Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form part of an ongoing study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by the Science and Technology Planning Project of Guangdong Province of China (nos. 2015B010128013, 2015B090911001, and 2015B090912001).

## References


[1] J. C. Hung and Y. H. Lee, "Intelligent travel information platform based on location base services to predict user travel behavior from user-generated GPS traces," *International Journal of Computers and Applications*, vol. 39, no. 3, pp. 155–168.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Proceedings of the Advances in Cryptology. CRYPTO 1985*, pp. 417–426, Santa Barbara, CA, USA, January 1986.

[3] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, p. 203, 1987.

[4] National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Publication, Gaithersburg, MD, USA, 2000.

[5] ANSI X9.63, *Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Key Transport Protocols*, American National Standards Institute, New Yark, NY, USA, 2000.

[6] Institute of Electrical and Electronic Engineers, NY, *P1363 Standard Specifications for Public Key Cryptography*, Institute of Electrical and Electronic Engineers, New Yark, NY, USA, 2000.

[7] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, p. 519, 1985.

[8] T. Güneysu and C. Paar, *Ultra High Performance ECC over NIST Primes on Commercial FPGAs*, Springer, Berlin, Germany, 2008.

[9] M. A. Nassar and L. A. A. El-Sayed, "Efficient interleaved modular multiplication based on sign detection," in *Proceeding of the 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–5, Marrakech, Morocco, November 2015.

[10] X. Hu, X. Zheng, S. Zhang, S. Cai, and X. Xiong, "A low hardware consumption elliptic curve cryptographic architecture over GF(p) in embedded application," *Electronics*, vol. 7, no. 7, p. 104, 2018.

[11] S. Ghosh, M. Alam, D. R. Chowdhury, and I. S. Gupta, "Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side channel attacks," *Computers & Electrical Engineering*, vol. 35, no. 2, pp. 329–338, 2009.

[12] K. Javeed and X. Wang, "FPGA based high speed SPA resistant elliptic curve scalar multiplier architecture," *International Journal of Reconfigurable Computing*, vol. 2016, Article ID 6371403, 10 pages, 2016.

[13] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 449–460, 2003.

[14] Z. Zhao and G. Bai, "Ultra high-speed SM2 ASIC implementation," in *Proceedings of the 2014 IEEE 13th International Conference*, pp. 182–188, Singapore, June 2014.

[15] G. Chen, G. Bai, and H. Chen, "A High-Performance elliptic curve cryptographic processor for general curves over GF(p) based on a systolic arithmetic unit arithmetic unit," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 5, pp. 412–416, 2007.

[16] D. N. Amanor, C. Paar, J. Pelzl, and V. Bunimov, "Efficient hardware architectures for modular multiplication," in *Proceedings of the International Conference on Field Programmable Logic and Applications, 2005*, Tampere, Finland, February, 2005.

[17] S. Ghosh, M. Alam, I. S. Gupta, and D. R. Chowdhury, "A Robust GF(p) parallel arithmetic unit for public key cryptography," in *Proceedings of the EUROMICRO DSD 2007*, pp. 109–115, Lubeak, Germany, August 2007.

[18] K. Javeed and X. Wang, "Low latency flexible FPGA implementation of point multiplication on elliptic curves over GF(p)," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 214–228, 2017.

[19] K. Javeed, X. Wang, and M. Scott, "High performance hardware support for elliptic curve cryptography over general prime field," *Microprocess & Microsystem*, vol. 51, pp. 331–342, 2016.

[20] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, "FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field," *IEEE Access*, vol. 7, Article ID 178811, 2019.

[21] Standard Specifications for Public-key Cryptography, "IEEE standard P1363," 2000, http://grouper.ieee.org/groups/1363.

[22] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, New York, NY, USA, 2004.