IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 1, 2019, accepted February 27, 2019, date of publication March 5, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903170

Speed-Oriented Architecture for Binary Field Point Multiplication on Elliptic Curves

JIAKUN LI[1], **SHUN'AN ZHONG**[1], **ZHE LI**[2], **SHAN CAO**[3], **JINGQI ZHANG**[1], **AND WEIJIANG WANG**[1]

[1]School of Information and Technology, Beijing Institute of Technology, Beijing 100081, China
[2]Bitmain Technologies Ltd., Beijing 100029, China
[3]Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Joint International Research Laboratory of Specialty Fiber Optics and Advanced Communication, Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China

Corresponding author: Weijiang Wang (wangweijiang@bit.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61801024.

ABSTRACT This paper introduces a novel speed-oriented architecture of point multiplication in elliptic curve cryptography. A balanced full-precision multiplier is proposed to shorten latency, and a new modular inversion architecture is integrated to reduce the total number of clock cycles in point multiplication. A modified Montgomery Ladder algorithm that takes three clock cycles to calculate one input bit is proposed to best utilize hardware resources. A mixed-pipeline technique is used to balance the delay of different paths and increase frequency. The proposed architecture is implemented on GF(2^{163}) and GF(2^{571}), based on Xilinx Virtex-5 and Virtex-7 FPGA. For GF(2^{163}), the design reaches 211 MHz, with 29309 LUTs, and 547 clock cycles or 2.6 μs latency on Virtex-5; 320.5 MHz, with 28911 LUTs and 1.7 μs latency on Virtex-7. For GF(2^{571}), the design reaches 186 MHz, with 286400 LUTs, and 1813 clock cycles or 9.6 μs latency on Virtex-5; 267 MHz, 290001 LUTs and 6.79 μs latency on Virtex-7. The proposed design achieves the lowest latency among all existing works, and its performance is also among the top. Furthermore, it is demonstrated that the proposed architecture maintains a high speed for larger binary fields, making it more suitable to be implemented in large-bit-length platforms with a higher security level. Since the multiplier and its segments work in different bit-length and refer to different fields, the proposed architecture can also be upgraded to a reconfigurable design to support multiple-field point multiplication in the future.

INDEX TERMS ECC (elliptic curve cryptography), point multiplication, ITA (Itoh Tsujii algorithm), Montgomery Ladder, FPGA implementation.

I. INTRODUCTION

The Elliptic Curve Cryptography (ECC) was proposed in 1985 by Koblitz [1] and Miller [2], respectively. Since then, a massive amount of scientific researches on ECC implementation have been carried out in succession, focusing on lower area consumption and higher efficiency. Compared to the presently prevalent public-key encryption system RSA, ECC achieves higher processing speed with shorter key length and less complex key certificates, while maintaining the same security level. Either prime field or binary field can be chosen as the implementation field of cryptographic mechanisms based on elliptic curves [3]. The prime field is more suitable for software implementation whereas binary field is more friendly to hardware implementation with its "carry-free" property.

With the explosive growth of Internet-based applications like peer-to-peer networks, ecommerce and distributed gaming, the demand for security in such systems has also grown markedly [4]. However, among these applications, some of the time-critical applications such as network servers where millions of heterogeneous client devices need to be connected, the processing speed of identity authentication needs to be further improved. In recent years, several ECC processor designs over binary field Koblitz curve have been proposed to increase encryption speed [5]–[9]. Since point multiplication (PM) is the main and the most time-consuming iterative operation during the whole encryption process, most researches focus on exploiting novel hardware architectures to speed up the operation process. Among these novel

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

high-speed architectures, the main strategies are to reduce the number of operating clock cycles by implementing parallel structures at the algorithm level, and to increase clock frequency by making proper use of pipeline stages to optimize critical path.

A. RELATED WORK

To increase the speed of point multiplication, the structure of modular multiplier is well-studied for its high logic complexity that dominates critical path. Two types of multipliers are usually implemented in high-speed/performance architectures, bit-parallel and digit-serial [10], [11].

Karatsuba-Ofman multiplier (KOM), as the representative of bit-parallel multipliers, can save resources by reducing the multiplicative complexity of two n-digit numbers from $O(n^2)$ to $O(n^{log_2 3})$ [12]–[14]. In a KOM, area reduction is traded with extra logic levels. Integrating a KOM in a PM system usually means to insert many pipelines to raise the operating frequency to an adequate number. This will result in larger multiplication latency as a sacrifice.

Full-precision multipliers, as the representative of digit-serial multipliers, can trade higher frequency with more area [15]–[20]. In these designs, although the multiplicative complexity remains $O(n^2)$, the number of critical logic levels is fewer than a KOM.

Apart from multipliers, various parallel structures have been exploited recently to reduce latency. In the proposed point multiplication designs [12]–[20], data flow was arranged ingeniously to satisfy complex data dependency while implementing parallel structures. In these designs, additional registers must be implemented across parallel multipliers to store intermediate values and break critical paths.

Recently, a fast point multiplication design based on a two-stage-pipeline KOM is proposed by [21]. When implemented on Xilinx Virtex-5, it takes 9470 LUTs and 4.6 μs (1363 clock cycles) to finish one operation, at the highest clock frequency of 294 MHz. Besides, two point multiplication architectures are proposed in [22] with quadratic full-precision multipliers. The first structure is called high-performance architecture, which is implemented with only one multiplier and takes 14202 LUTs and 1119 clock cycles at the clock frequency of 352 MHz on Xilinx Virtex-7. The other one, called low-latency architecture, is implemented with three multipliers and takes 41090 LUTs and 450 clock cycles at the clock frequency of 159 MHz on Xilinx Virtex-7. Due to the parallelization, the low-latency architecture is able to process one PM operation in 2.83 μs, which outperforms all previous works. But this architecture has much optimization space specifically in that the operating frequency drops more than 50% when the chosen finite filed gets larger. Moreover, the modular inversion operation which is only executed once in a PM operation takes nearly 1/4 of the total clock cycles. Since parallel multipliers can only reduce the clock cycles of Montgomery Ladder process [see Sec II.C], the clock cycles to do modular inversion (or final coordinate transformation) have become the dominating factor. Consequently, to meet the growing demand for speed in ECC applications, a well designed architecture with both high speed and robustness is urgently needed.

The proposed design is implemented on Field-Programmable Gate Array (FPGA) platform to evaluate its speed and performance. The reasons are as follows. First, FPGA is flexible and new designs can be instantly upgraded through download. Second, the previous results over binary field Koblitz curve are mostly FPGA results. More data can be added to comparison. Last, former FPGA results mainly fall into two series of devices, Xilinx Virtex-5 and Virtex-7. The comparison can therefore be done fairly, regardless of the difference in manufacture procedure and PVT (process, voltage and temperature).

B. MAIN CONTRIBUTION

In this paper, we aim at designing a high-speed point multiplication architecture with among-the-top performance. An architecture with two novel balanced full-precision modular multipliers which have quadratic multiplicative complexity is proposed. The Montgomery Ladder unit shares one of the multipliers with the modular inversion unit to reduce area. A modified modular inversion architecture is proposed to optimize critical path and shorten latency. As a result, the whole design achieves the processing speed of 1.7 μs per PM operation at the frequency of 320.5 MHz based on Xilinx Virtex-7 over $GF(2^{163})$. Implementation result shows that the processing speed and performance of our speed-oriented point multiplication architecture are higher than all previous works. Thus, our design is of great significance for high-speed ECC applications.

The main contributions of this paper are as follows:

1) A novel balanced full-precision multiplier that combines bit-parallel and digit-serial multipliers is proposed. It maintains high operating frequency over both $\mathrm{GF}(2^{163})$ and $\mathrm{GF}(2^{571})$.
2) A modified three-clock-cycle (3CC) Montgomery Ladder Algorithm is proposed. The number of operation clock cycles is reduced to the minimum. The design is able to process at the speed of three clock cycles per bit and the total latency is the shortest among all existing works.
3) A corresponding mixed-pipeline architecture with two multipliers working in parallel is proposed to achieve high speed. Integrating different-stages pipelines to balance delay of different datapaths has efficiently increased system frequency.
4) A high-speed modular inversion unit is designed to work with our point multiplication architecture. By sharing the multiplier of Montgomery Ladder process, we manage to reduce the additional area brought by modular inversion to the minimum. The critical path of inversion unit is also optimized to fit the clock frequency of the point multiplication architecture. The number of operation clock cycles is reduced to the minimum.

5) To reduce complexity and area of modular inversion unit, the modular powering units are simplified into one-level logic and high powering units are eliminated. As a result, the operating frequency is improved by 23% with similar amount of LUTs.

The rest of the paper is organized as follows. Section II introduces the basic point multiplication algorithms over GF(2^m). Section III presents the proposed modular multiplier, and the proposed architecture of point multiplication is introduced in Section IV. Section V compares the results between the proposed architecture and other work. Section VI concludes the paper.

II. POINT MULTIPLICATION ALGORITHM OVER EXTENDED BINARY FIELDS

A. BASIC OPERATIONS OVER EXTENDED BINARY FIELDS

Three common elliptic curves defined over binary field were given extensive attention over these years: Binary Edwards curve(BEC), Binary Huff curve(BHC) and Binary Koblitz curve(BKC). BEC and BHC are recent-proposed curves with algorithm-level preventive of Side Channel Attacks (SCAs) brought by unified addition laws [23]–[25]. BKC is a standard curve that requires less computational cost than BEC and BHC. With the help of Montgomery Ladder, encryption over BKC can reach the same level of resistance over SCAs. Hence ECC implementation based on this curve is expected to be efficient and secure.

The five basic operations on GF(2^m) are modular addition, modular reduction, modular powering, modular multiplication and modular inversion. The operations of ECC over GF(2^m) are based on the BKC known as $E : y^2 + xy = x^3 + ax + b$. The carry-free property on GF(2^m) makes all basic operations simple for hardware implementation. Modular addition together with modular subtraction is merely bitwise Exclusive OR (XOR) operation of two inputs. Modular reduction requires an irreducible polynomial to reduce $(2m - 1)$-bit results of modular multiplication and modular squaring back to an m-bit value. For example, in GF(2^{163}), the polynomial recommended by National Institute of Standards and Technology, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ is used to ensure the result is within the same binary field. Modular powering and multiplication follow the same rule of operation. By replacing binary addition in normal squaring and multiplication with modular addition, and reducing the results with an irreducible polynomial, modular squaring and multiplication are derived. Modular inversion is the inverse operation of modular multiplication.

The hardware implementation of modular inversion is based on Fermat's Little Theorem which can transform a modular inversion operation into several modular multiplication operations and modular powering operations in the given extended binary field.

On extended binary field GF(2^m), Fermat's Little Theorem can be described by

$$x^{-1} = x^{2^m-2} = x^{2(2^{m-1}-1)}$$
$$= x^{2(2^{m-2}+\ldots+2^1+2^0)} \ (mod\ 2^m). \tag{1}$$

Considering the powering term $2^{m-2} + \ldots + 2^1 + 2^0$, if m is odd, it can be simplified as

$$2^{m-2} + \ldots + 2^0 = (2^{\frac{m-1}{2}-1} + \ldots + 2^0)(2^{\frac{m-1}{2}} + 1) \tag{2}$$

And if m is even, it will be

$$\begin{aligned}2^{m-2}+\ldots+2^0 &= 2(2^{m-3} + \ldots + 2^0) + 1\\ &= 2(2^{\frac{m-2}{2}-1} + \ldots + 2^0)(2^{\frac{m-2}{2}} + 1) + 1\end{aligned} \tag{3}$$

In this manner, modular inversion over GF(2^{163}) can be transformed to

$$\begin{aligned}x^{-1} &= x^{2^{163}-2} = x^{2(2^{162}-1)}\\ &= x^{2(2^{81}+1)\{2(2^{40}+1)(2^{20}+1)(2^{10}+1)(2^5+1)[2(2^2+1)(2+1)+1]+1\}}\end{aligned} \tag{4}$$

This technique shown by Eqn.(2)-(4) is called Itoh Tsujii algorithm (ITA). In the algorithm, inversion operation over GF(2^m) can be simplified into two basic operations. The first operation is to calculate $x^{(2^n+1)a}$ when the input is x^a, and the second operation is to calculate x^{2a+1} when the input is x^a and x. The complex modular inversion operation can be done by scheduling these two basic operations, which are different combinations of two field operations modular multiplication and modular powering. Therefore the performance of modular inversion unit mainly depends on modular multipliers and modular powering units.

B. THE MONTGOMERY LADDER ALGORITHM

On a given elliptic curve, based on a base point P and an integer k, the point multiplication process can be computed by k times point additions, as $Q = P + P + P + +P = kP$.

To enhance resistance to Side Channel Attacks (SCAs) and be more friendly to hardware implementation, the Montgomery Ladder, a commonly-applied point multiplication algorithm for ECC is considered in this paper. Based on the Montgomery Ladder, point multiplication can be transformed into operations of point addition and point doubling. When processing k, the Montgomery Ladder requires one point addition and one point doubling whether k_i is 0 or 1. This unification against different patterns of k prevents the system from the harm of Simple Power Attacks (SPAs), one form of SCAs.

However, in affine coordinate system, point addition and point doubling need one modular inversion operation each time. Since modular inversion is the most time-consuming operation among all basic operations, we transform the process of point multiplication from affine coordinate system to projective coordinate system. As a result, most modular inversion operations are eliminated and there only leaves one inversion at the end of the whole point multiplication process. Meanwhile, during the Montgomery Ladder over projective coordinate system, only calculation of X-coordinate (X and Z) is a must and final value of Y-coordinate can be recovered in the end. This special property makes the algorithm more efficient to implement on hardware. Given two points

TABLE 1. Complexity type and hardware resources of different multipliers over $GF(2^m)$.

Works	Type	#XOR	#AND	Critical Path (T_{mul})
[26]	Quadratic	m^2-1	m^2	$T_A+(1+log_2 m)T_X$
[26]	Subquadratic	$5.5m^{log_2 3}-3m+0.5$	$m^{log_2 3}$	$T_A+(3+2log_2 m)T_X$
[27]	Pipelined-Quadratic	m^2+3m	m^2	$T_A+((m/p)+log_2 m)T_X$
[22]	Pipelined-Quadratic	$m(m-1)+(n-1)m+(r-1)m$	m^2	$T_A+(log_2(m/n)+n+2r)T_X$

Algorithm 1 6CC Montgomery Ladder Algorithm When $k_i = 1$

```
Require: $P(X_1, Z_1)$, $Q(X_2, Z_2)$, $k_i = 1$ and $k_{i+1}$
Ensure: Point Addition and Point Doubling
  for $i = t-2$ downto 0 do
    if ($k_{i+1} = 1$) then
      st1: $Z_1 \leftarrow X_2Z_1; A \leftarrow Z_2$
      st2: $X_1 \leftarrow X_1Z_2; Z_2 \leftarrow A^2; R_2 \leftarrow A^4; A \leftarrow X_2$
    else if ($k_{i+1} = 0$) then
      st1: $Z_2 \leftarrow X_1Z_2; A \leftarrow Z_2$
      st2: $X_2 \leftarrow X_2Z_1; Z_2 \leftarrow A^2; R_2 \leftarrow A^4; A \leftarrow X_2$
    end if
      st3: $X_2 \leftarrow bR_2 + A^4; R_1 \leftarrow A^2$
      st4: $Z_2 \leftarrow R_1Z_2; A \leftarrow X_1 + Z_1$
      st5: $X_1 \leftarrow X_1Z_1; Z_1 \leftarrow A^2$
      st6: $X_1 \leftarrow xZ_1 + X_1$
  end for
  conversion step for $y$
```

$P_1(X_1, Y_1, Z_1)$ and $P_2(X_2, Y_2, Z_2)$ on the elliptic curve, point addition is $P_3 = P_1 + P_2$. And value of P_3 follows:$X_3 = x_P(X_1Z_2 + X_2Z_1)^2 + X_1X_2Z_1Z_2$ and $Z_3 = (X_1Z_2 + X_2Z_1)^2$. Point doubling is $P_3 = 2P_1$, which follows $X_3 = X_1^4 + bZ_1^4$ and $Z_3 = X_1^2Z_1^2$.

To better utilize hardware resources when implementing the Montgomery Ladder, a hardware-based algorithm is proposed by [22]. As is shown in Algorithm 1, it uses one modular multiplier in six consecutive clock cycles to compute one bit of k. Based on the relations shown above, we need at least six multiplication operations to calculate P_3. In this algorithm, with one multiplier, six multiplication operations are executed in six different clock cycles. Therefore, this algorithm achieves the most efficient time utilization and theoretically the minimum number of computation clock cycles for one multiplier. Moreover, from the Montgomery Ladder Algorithm, operations for $k_{i+1} = 0$ and $k_{i+1} = 1$ are similar in a way. Thus, some datapaths can be shared and others can be built in the same way, which tremendously simplifies the whole design.

III. DESIGN OF MODULAR MULTIPLIER

In the point multiplication architecture, more than 90% hardware resources are consumed by modular multipliers. In the meanwhile, modular multipliers also dominate the total latency and processing speed. High-speed multipliers mostly fall into the category of bit-parallel KOM and digit-serial full-precision. On prime field, previous researches revealed KOM is more balanced. One specific design [11] shows that the modular reduction can be done along with the multiplication in a KOM, to further reduce the resources. On binary field, since modular reduction is simply XOR, there is no need for sharing reduction with the design of multiplication. The resource saving of KOM is not as efficient, so these two multipliers each has appropriate applications. In this section, we will first make comparison of the proposed multiplier and the previous ones on binary field, and then present parallelization of multipliers to implement a high-speed architecture.

A. PREVIOUS ARCHITECTURES OF MODULAR MULTIPLIER

As is mentioned in Sec.II.A, modular multiplication is similar with normal multiplication. There are only two differences: 1) carry-free addition and 2) modular reduction. Let $f(x)$ be an m-degree irreducible polynomial and two operands $A(x) = \sum_{i=0}^{m-1} a_ix^i$ and $B(x) = \sum_{i=0}^{m-1} b_ix^i$. A modular multiplication denotes $C(x) = A(x)B(x) \; mod \; f(x)$. Main complexity of this procedure lies in polynomial multiplication, which is directly shown as:

$$D(x) = A(x)B(x) = \sum_{k=0}^{2m-2} d_kx^k$$

where

$$d_k = \sum_{i+j=k} a_ib_j, \qquad 0 \le i, \; j \le m-1$$

Though the complexity of polynomial multiplication is $O(n^2)$ [8], it is rather costly to implement this part directly, especially when n is a large number such as 163, 283 or 571. Thus, many previous works focused on optimizing the polynomial multiplication to speed up modular multiplication.

Basically, there are two ways to design modular multipliers: Karatsuba-Ofman multipliers (KOM) and quadratic digit-serial full-precision multipliers and the comparison between them is shown in Tab. 1. The KOM is a subquadratic design which reduces the multiplicative complexity of two n-digit numbers from $O(n^2)$ to at most $O(n^{log_2 3})$ by optimizing the algorithm of polynomial multiplication. In quadratic full-precision multipliers, the logic levels are reduced but the complexity remains $O(n^2)$. Since the number of pipeline stages inserted in the multiplier is determined by the proposed structure, it is only fair to compare these two types of multipliers with the same number of pipeline stages. In this case, the KOM multiplier trades reduced area with worse critical path, while the full precision multiplier results in less critical delay and increased area. In the theoretical analysis of quadratic and subquadratic multipliers [22], [26],

Algorithm 2 Algorithm of Balanced Full-Precision Multiplier

```
Require: m-bit $a$, m-bit $b$
Ensure: $a * b$
    Given $a = a_1 2^{m-2} + a_2, b = b_1 2^{m-2} + b_2$
    Thus,
    $a * b = (a_1 2^{m-2} + a_2) * (b_1 2^{m-2} + b_2)$
          $= a_1 * b_1 * 2^m + (a_2 b_1 + a_1 b_2) * 2^{m/2} + a_2 * b_2$
```

though quadratic multipliers costs 2.56 times more hardware resources than the subquadratic ones, the quadratic multipliers are two times faster. Therefore, in a speed-oriented design, quadratic full-precision multipliers are considered as a better choice.

Khan and Benaissa [22] proposed a segmented full-precision multiplier which has quadratic complexity. However, the point multiplication architecture only performs well on GF(2^{163}) with one segmented full-precision multiplier at the operating frequency of 228 MHz. In other cases, such as on GF(2^{571}) or with three multipliers working in parallel, the frequency decreases by nearly half. In our design, we optimize the critical path and propose a balanced full-precision multiplier, with which our point multiplication architecture maintains high operating frequency on different extended binary fields.

B. PROPOSED BALANCED FULL-PRECISION MULTIPLIER (BMUL)

The proposed balanced full-precision multiplier (BMUL) consists of four segmented full-precision sub-multipliers which are half the designated size of field multiplication. The multiplication of two m-bit inputs can be done by using four multiplications with the input size of m/2 bits when they are organized as Algorithm 2:

Fig. 1 shows the architecture of the proposed BMUL, and Fig. 2 shows the architecture of the four segmented full-precision sub-multipliers implemented in the proposed BMUL. An m-bit multiplicand a and a multiplier b are divided equally into two (m/2)-bit inputs a_1, a_2, and b_1, b_2 respectively. Each two of the four partitioned inputs are then strobed to four segmented full-precision sub-multipliers and after one pipelined stage, four partitioned multiplication results are obtained. The results are then shifted, XORed (added in extended binary field) and reduced to get final modular multiplication result.

The proposed BMUL is partitioned based on the same idea with Karatsuba multiplier, but the partitioning is merely on the top level. Increasing the number of partition levels and sharing sub-multipliers can reduce area, however it will also increase the total latency. Meanwhile, another difference lies in the number of partitioned multipliers on one partition level. In a normal Karatsuba multiplier, a_1 and a_2, b_1 and b_2 are XORed and then strobed to the partitioned multiplier. Consequently, the output of this multiplier is

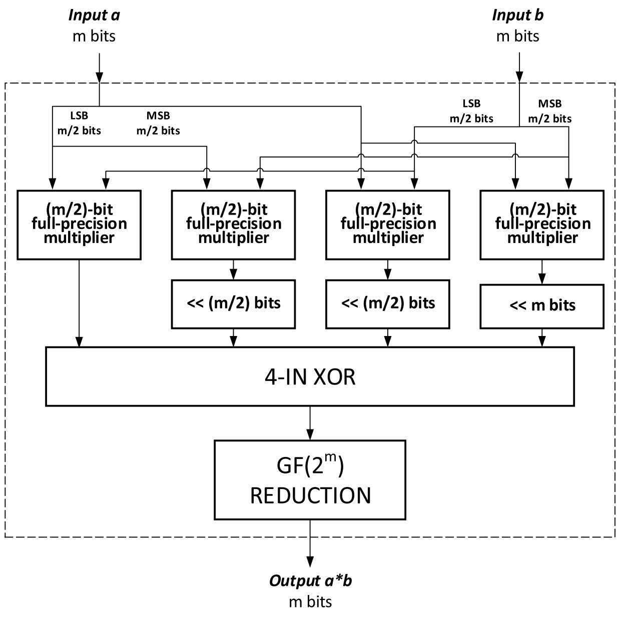

FIGURE 1. The proposed architecture of balanced full-precision multiplier (BMUL) with 4 half-sized segmented full-precision multipliers.

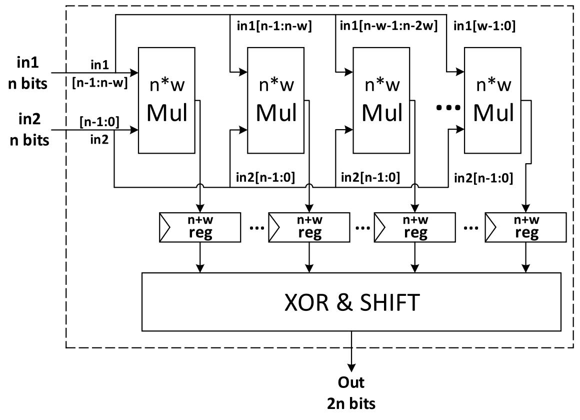

FIGURE 2. The architecture of segmented full-precision multiplier.

$(a_1 + a_2) * (b_1 + b_2) = a_1 * b_2 + a_2 * b_1 + a_1 * b_1 + a_2 * b_2$, while the outputs of the other two multipliers are $a_1 * b_1$ and $a_2 * b_2$. Then, after a simple XOR operation, $a_1 * b_2 + a_2 * b_1$ is obtained. It will then go through the SHIFT-and-XOR module with $a_1 * b_1$ and $a_2 * b_2$ to get the final result. This technique can potentially eliminate one of the partitioned multipliers and reduce area. However, the two more levels of XOR on the input and output of the partitioned multipliers will become two more levels of LUT in FPGA implementation which results in a drop of frequency.

Therefore, the BMUL takes up similar area with full-precision multiplier, and achieves higher frequency. Moreover, a potential advantage of BMUL is that it can work as a reconfigurable multiplier with modular reduction units for different extended binary fields. Four partitioned sub-multipliers can serve as multipliers for another binary field. For instance, if the main multiplier works on GF(2^{571}), the partitioned multipliers can work on GF(2^{283}). In a reconfigurable ECC design, the multipliers of different binary fields can be shared in this way. Since multiplier usually takes up more than half of the area, an reconfigurable multiplier can potentially reduce area and increase frequency, solve the conflict between area and speed in a reconfigurable ECC design.

The proposed BMUL has one pipeline stage inside, and the outputs of multipliers are strobed back to its inputs. This loop ensures that the placing of the registers inside the multiplier will not affect the length of critical path. Therefore, we place them where the combination logic is most complex, so that it can simplify routing logic and eliminate unnecessary influence on clock frequency.

C. THE PARALLELIZATION OF MODULAR MULTIPLIERS

Based on the 6CC algorithm (which requires only one multiplication each clock cycle), various recent designs focused on proposing new implementation with only one modular multiplier [21], [22]. Despite their high clock frequency, these designs suffer from the problem of excessive computational clock cycles, as a result of the algorithm itself. For instance, on GF(2^{163}), the total computational clock cycles of the Montgomery Ladder is at least $6 \times 163 = 978$ clock cycles, which does not include the clock cycles to do the final coordinate transformation. Such speed becomes increasingly incompetent for today's encryption speed requirement.

One common way to increase speed at the cost of area is parallelization. Compared with increasing the number of the whole point multiplication architecture, increasing the number of multipliers can reduce the area of design by sharing other operation modules. Consequently, the area of the design will not double, while the latency drops by half, which results in a faster, and more balanced performance.

In our research, the ideal number of parallel multipliers is experimented to bring out the best performance. In order to fully utilize multiplication resources, the number of multipliers has to be a factor of 6 (which refers to 6CC algorithm), leaving the options of 1, 2, 3 and 6 multipliers. Actually, six multipliers cannot work simultaneously because six multiplications in the 6CC algorithm have crossover data dependency. Our research covers the other three options. Designs with one multiplier are well-studied in recent years and suffer from latency problems. On the bright side, the data dependency can support up to two stages pipelines to be inserted in the loop, cutting the critical path into three parts. These designs usually have higher frequency than the others. Design with three multipliers can reduce latency to almost 1/3. However, the frequency becomes the limit of its speed. With only one stage of pipeline breaking the critical path into two, complex data dependency means more cascaded multiplexers making the critical delay even longer. Design based on two multipliers gather both virtues of frequency and latency. The parallelization halves the number of clock cycles, and the sharing of modules reduces the area from simply doubling. Although data dependency of two multipliers can only allow one pipeline to be inserted into the critical path, a mixed-pipeline architecture can be implemented to further increase frequency. Specifically, the most critical path is realized with the least logic and cut in half, and other paths are built with two pipeline stages. Results show that the mixed-pipeline two-multiplier architecture can be even faster than a three-multiplier architecture, and equally balanced to most one-multiplier architectures.

IV. ARCHITECTURE OF POINT MULTIPLICATION IMPLEMENTATION

In point multiplication, there are mainly two parts: the Montgomery Ladder unit and modular inversion unit. Fig. 3 shows the overall architecture of the proposed point multiplication design. Apart from combining the architecture of 3CC Montgomery Ladder and modular inversion, an additional modular adder and some extra DFFEs (D Flip Flops with Enable signal) are implemented to finish extra calculation required during final coordinate transformation. The Montgomery Ladder unit is in charge of calculating point addition and point doubling cyclically based on an input k and it will output results in projective coordinate system. By modular inversion unit, results in affine coordinate system can be calculated in the end.

In this section, we first propose our novel 3CC Montgomery Ladder algorithm and then analyze data dependency in the algorithm. Based on the data dependency, the corresponding data flow is designed which is illustrated in detail. The data flow proves the feasibility of our novel 3CC Montgomery Ladder architecture. From the data flow, a novel hardware design is proposed and the mixed-pipeline architecture is introduced to explain the key to frequency improvement. Then, a novel high-speed architectures of modular inversion is introduced with the improvement to previous modular inversion architectures.

A. PROPOSED ARCHITECTURE WITH 3CC MONTGOMERY LADDER ALGORITHM BASED ON TWO PARALLEL MULTIPLIERS

1) THE 3CC MONTGOMERY LADDER

Based on the 6CC Montgomery Ladder in Section II.C, the minimum number of clock cycles to process one bit of the input k with two parallel multipliers is 3 clock cycles. In 6CC Montgomery Ladder Algorithm, since there is no data dependency between st1 and st2, st3 and st4, st5 and st6, the multiplications can be distributed to two multipliers. The proposed algorithm for two parallel multipliers is called 3CC Montgomery Ladder, as is shown in Algorithm 3.

2) DATA DEPENDENCY OF THE 3CC MONTGOMERY LADDER

According to Algorithm 3, the difference in operation between $k_i = 1$ and $k_i = 0$ mainly lies in the order of the input. In other words, the algorithm for $k_i = 0$ can be obtained by exchanging the X_1 and X_2, Z_1 and Z_2 in the algorithm for $k_i = 1$. Therefore, data dependency can be divided into three

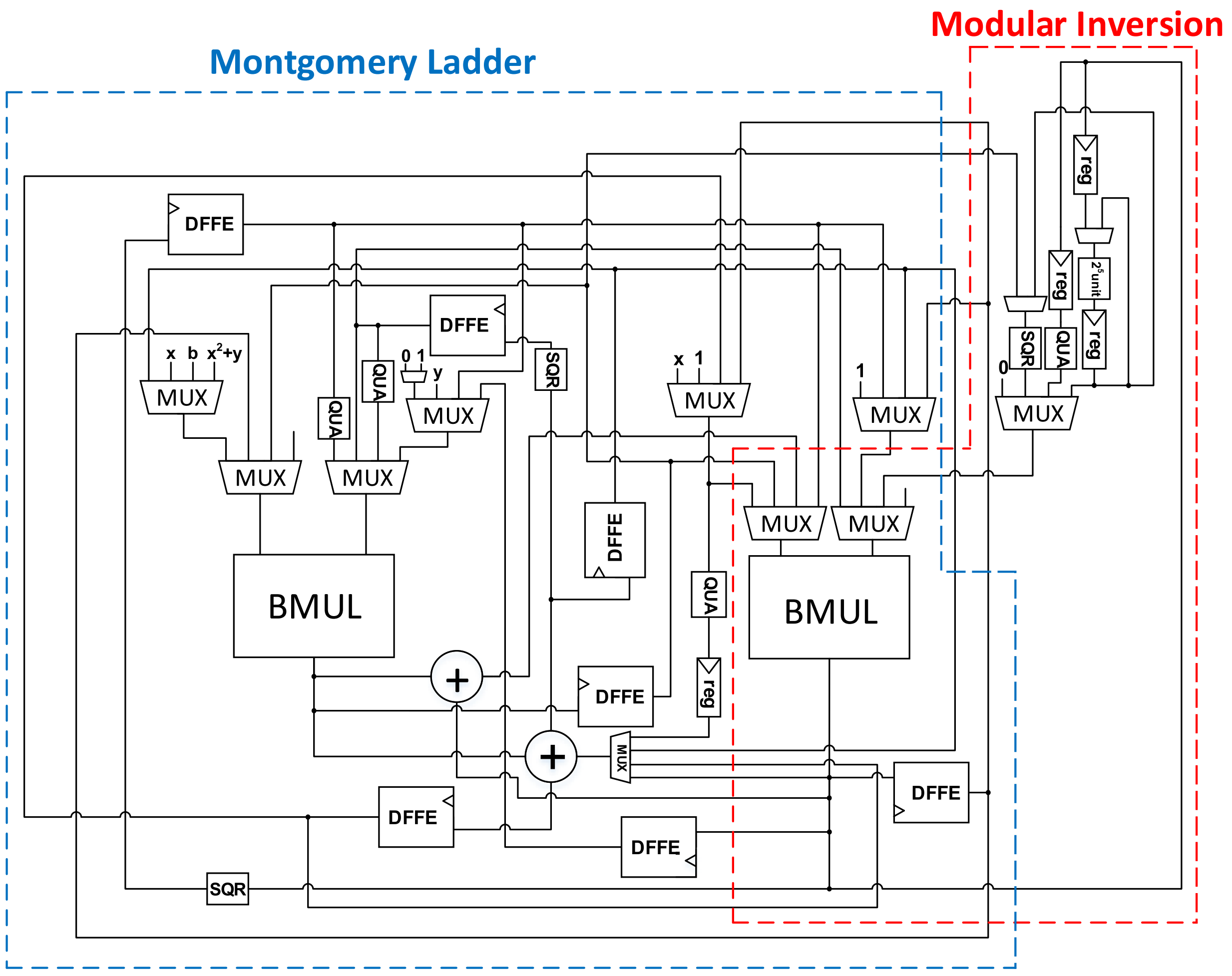

FIGURE 3. Proposed point multiplication architecture including 3CC Montgomery Ladder unit and Modular inversion unit.

Algorithm 3 Proposed 3 CC Montgomery Ladder Algorithm

```
Require: $P(X_1, Z_1), Q(X_2, Z_2), k_i$
Ensure: Point Addition and Point Doubling
  for $i = t - 2$ downto 0 do
    if $(k_i = 1)$ then
      st1: $Z_1 \leftarrow X_2Z_1; X_1 \leftarrow Z_2X_1$
      st2: $X_2 \leftarrow bZ_2^4 + X_2^4; Z_2 \leftarrow Z_2^2X_2^2$
      st3: $X_1 \leftarrow X_1Z_1 + x(X_1 + Z_1)^2; Z_1 \leftarrow (X_1 + Z_1)^2$
    else if $(k_i = 0)$ then
      st1: $Z_2 \leftarrow X_1Z_2; X_2 \leftarrow Z_1X_2$
      st2: $X_1 \leftarrow bZ_1^4 + X_1^4; Z_1 \leftarrow Z_1^2X_1^2$
      st3: $X_2 \leftarrow X_2Z_2 + x(X_2 + Z_2)^2; Z_2 \leftarrow (X_2 + Z_2)^2$
    end if
  end for
  conversion step for $y$
```

groups: data dependency of k_i, data dependency between the same k_i and k_{i-1} and data dependency between different k_i and k_{i-1}. Specific data dependency is as follows : for k_i, the input of st3 depends on the output of st1; between same k_i and k_{i-1}, the input of st1 for k_{i-1} depends on the output of st3 for k_i, and the input of st2 for k_{i-1} depends on the output of st2 for k_i; between different k_i and k_{i-1}, the input of st1 for k_{i-1} depends on the output of st2 for k_i, and the input of st2 for k_{i-1} depends on the output of st3 for k_i. Among them, the critical one is the dependency of st1 input for k_{i-1} on the st3 output for k_i, leaving only one clock cycle to finish the operation. Therefore it is determined by the 3CC algorithm that only one pipeline stage can be inserted in the most critical data-path.

3) DATA FLOW

To better show how hardware units work orderly to fit with the 3CC Montgomery Ladder Algorithm, data flow of the whole architecture will be elaborated in this subsection.

The data flow, which is related to both k_i and k_{i-1}, consists of four different situations, $k_i = 0$ and $k_{i-1} = 0$, $k_i = 1$ and $k_{i-1} = 1$, $k_i = 1$ and $k_{i-1} = 0$, $k_i = 0$ and $k_{i-1} = 1$. Since the difference between $k_i = 1$ and $k_i = 0$ is merely an exchange of X_1 and X_2, as well as Z_1 and Z_2, we will only

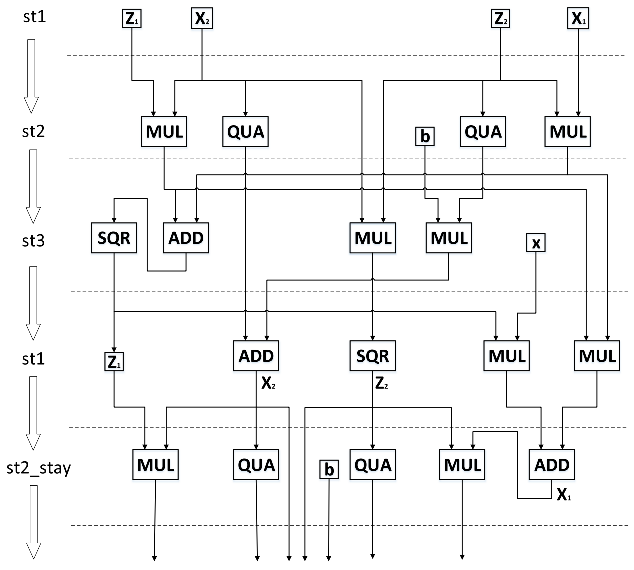

FIGURE 4. Data flow of $k_i = k_{i-1} = 1$.

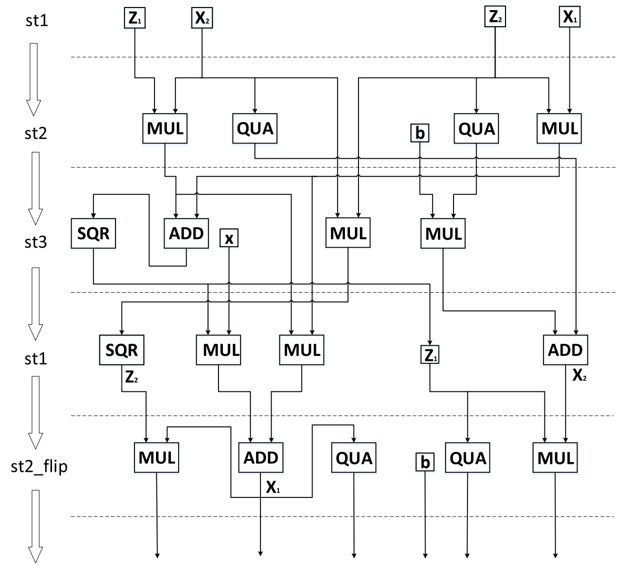

FIGURE 5. Data flow of $k_i = 0$ and $k_{i-1} = 1$.

discuss the data flow in one case ($k_{i-1} = 1$), and the data flow when $k_{i-1} = 0$ can be easily obtained by exchanging the input. The situations of $k_i = 1$ while $k_{i-1} = 1$, and $k_i = 0$ while $k_{i-1} = 1$ are shown respectively, in Fig. 4 and Fig. 5.

It can be gathered from the figure that the two displayed situations are completely unified against potential SPAs. A control unit is responsible for selecting each MUX and enabling the DFFEs to store the intermediate computation results. By sharing of states in different situations of k_i and k_{i-1}, the control unit is also simplified to save area.

4) PROPOSED ARCHITECTURE OF MONTGOMERY LADDER

Based on the data flow of 3CC Montgomery Ladder, we present the architecture as is shown in Fig. 6. Two modular squaring units (SQR in Fig. 6), three modular 2^2th powering units (QUA in Fig. 6), one modular addition unit and two balanced full-precision multipliers are included. Compared to the one-multiplier implementation [22] which utilizes one modular squaring unit, one modular 2^2th powering unit, two modular addition units and one full-precision multiplier, the proposed architecture doubles the number of multipliers and modular square units, implements more than twice the number of 2^2th power unit, and reduces the number of modular adder, so the area of the main operation units is less than twice of the one-multiplier design. Furthermore, the 3CC algorithm introduces less routing logic, resulting in an extra drop in area.

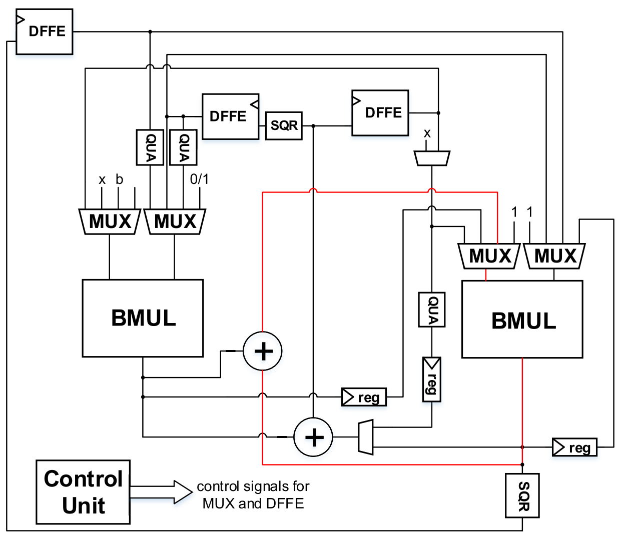

FIGURE 6. Proposed architecture of 3CC Montgomery Ladder.

A novel mixed-pipeline architecture is implemented in the proposed design as the key factor of frequency increase. The architecture takes advantage of the special data dependency of the 3CC Montgomery Ladder. Only the most critical data path (the red path in 8) is designed to be one-stage-pipelined, while the other paths are two-stage-pipelined. The logic on the critical path is reduced. For instance, the modular adder is duplicated to be implement individually so that the MUX before adder will not be on the critical path, and the fan-out of the adder is reduced to improve timing. Since modular adder is merely m-bit XORs on $GF(2^m)$, the area increment is negligible, while the timing improvement is critical. The critical path delay is reduced to the delay of MUX, modular adder, plus the delay of modular multiplier. Specifically, $critical\ path = t_{4-1\ mux} + t_{add} + t_{mul}$.

B. ARCHITECTURE OF MODULAR INVERSION

Compared to other basic operations in extended binary fields, modular inversion is a very costly operation. Fortunately, based on Fermat's Little Theorem, ITA can finish the modular inversion operation through modular multiplication and modular powering, which makes it easier for hardware implementation.

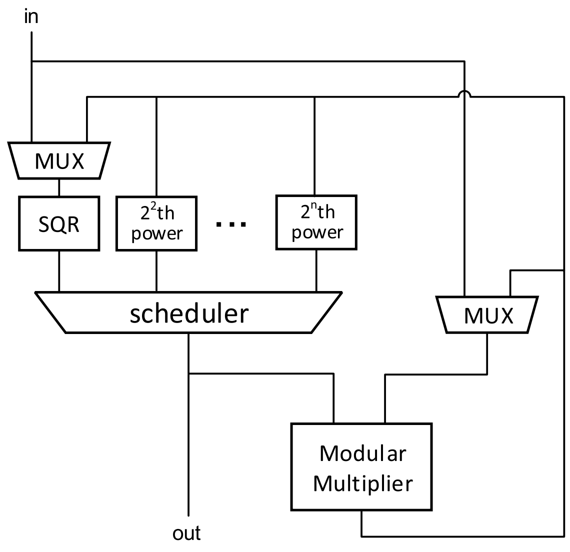

FIGURE 7. Common modular inversion architecture.

The former works mainly aimed at either low latency or low area. In pursuit of low area, cascaded modular 2^nth powering units are used to reduce the complexity, and hybrid Karatsuba multiplier is used to reduce area in [28]. Parallel cascades of squaring units and square-root units are used to reduce area utilization in [29]. In pursuit of low latency, 2^n units of combinational logic are used to speed up the calculation of powering at a large cost of area, which reaches the minimum theoretical clock cycles of one operation in [30]. A tree-structure-k-times squarer block is implemented in the ITA structure to achieve high clock frequency in [27].

However, none of the above architectures is suitable to implement in a complete point multiplication process. Since the inversion unit has to share multiplier with the Montgomery Ladder unit to reduce area, modular inversion unit and the Montgomery Ladder unit have to work at the same frequency. In other words, area, computational clock cycles as well as operating frequency need to be optimized at the same time.

To implement ITA, a common architecture is used as shown in Fig. 7. The differences of existing designs mainly lie in powering units, design of multiplier and pipeline stages inserted in the loop. In our case, since the modular multiplier is shared by the whole design, the extra area brought by modular inversion depends on the modular powering units. According to [31], if we simplify the modular powering units into XOR gate arrays, the area of modular powering grows exponentially as powering number increases. Therefore, elimination of higher powering units can effectively reduce the area. However, higher powering operations can only be achieved by reusing lower powering units for several clock cycles. The implementation area is reduced by compensating with clock cycles. For instance, in GF(2^{163}), the modular powering units required to finish modular inversion in the minimum clock cycles are squaring, 2^2th powering, 2^5th powering, 2^{10}th powering, 2^{20}th powering, 2^{40}th powering and 2^{81}th powering. If 2^{81}th powering unit is excluded, 2^{81}th powering operation has to be replaced by going through 2^{40}th powering unit twice (modular squaring can be attached to the output of 2^{40}th powering unit to reduce one clock cycle). In total, the latency increases by one clock cycle, while the area drops massively. If 2^{40}th powering unit is also excluded, the total latency increases by five clock cycles, while the area drops more than 50%. These two modifications seem hostile to latency, but it is very beneficial to optimize the critical path as well as reduce area. In fact, if both 2^{40}th powering and 2^{81}th powering are eliminated, the critical path will be much shorter which is close to the critical path of the Montgomery Ladder. If they are reserved, three to four pipeline stages must be added to separate the complex combination logic, which will result in three to four times latency and more area consumption.

Algorithm 4 Hardware ITA Algorithm

```
Require: $x$
Ensure: $z = x^{-1}$
  st1 :$z \leftarrow x^2$           (squaring unit in $GF(2^{163})$)
  st2 :$y \leftarrow x \bullet z$       (multiplication in $GF(2^{163})$)
  st3 :$z \leftarrow y^{2^2}$          ($2^2$ unit in $GF(2^{163})$)
  st4 :$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st5 :$z \leftarrow y^2$           (squaring unit in $GF(2^{163})$)
  st6 :$y \leftarrow x \bullet z$       (multiplication in $GF(2^{163})$)
  st7 :$z \leftarrow y^{2^5}$          ($2^5$ unit in $GF(2^{163})$)
  st8 :$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st9 :$z \leftarrow y^{2^{10}}$         ($2^5$ unit in $GF(2^{163})$ for 2 clock cycles)
  st10:$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st11:$z \leftarrow y^{2^{20}}$         ($2^5$ unit in $GF(2^{163})$ for 4 clock cycles)
  st12:$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st13:$z \leftarrow y^{2^{40}}$         ($2^5$ unit in $GF(2^{163})$ for 8 clock cycles)
  st14:$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st15:$z \leftarrow y^2$           (squaring unit in $GF(2^{163})$)
  st16:$y \leftarrow x \bullet z$       (multiplication in $GF(2^{163})$)
  st17:$z \leftarrow y^{2^{81}}$         ($2^5$ unit in $GF(2^{163})$ for 16 clock cycles)
  st18:$y \leftarrow y \bullet z$       (multiplication in $GF(2^{163})$)
  st19:$z \leftarrow y^2$           (squaring unit in $GF(2^{163})$)
  return $z$;
```

Fig. 8 elaborates the proposed architecture for modular inversion in point multiplication, and Algorithm 4 shows the corresponding hardware algorithm (for data path controlling). With a full consideration on timing, latency, and area, we choose to reserve modular squaring unit, 2^2th powering unit and 2^5th powering unit. Several pipeline stages are inserted as shown to make sure the critical path of modular inversion unit is close to that of the Montgomery Ladder (It is predictable that more stages are required if more powering units are reserved.). The total area of the powering units is less than 1000 LUTs, which is nearly negligible to the whole design. Although the latency increases to nearly 50 cycles by the elimination of higher powering units, the latency cost is bearable. As is discussed in Section II.B, in a whole point

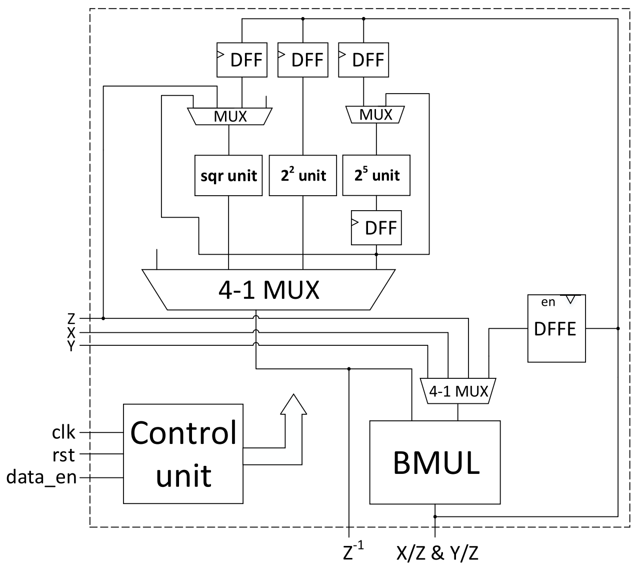

FIGURE 8. Proposed architecture of modular inversion in point multiplication.

multiplication process, modular inversion is only used once, at the coordinate transformation stage after the Montgomery Ladder. For example in GF(2^{163}), modular inversion takes up to 52 clock cycles. Compared to 489 clock cycles that the Montgomery Ladder costs, the additional clock cycles brought by inversion is nearly negligible, and is worthy to trade apparent improvement on frequency and area consumption. Our design of modular inversion is also a key factor to the outstanding overall performance.

C. LATENCY CALCULATION

The whole process of one point multiplication is to first calculate the Montgomery Ladder and get the result in projective coordinate X and Z. Since y_p is not used in this process, an extra coordinate transformation is needed to recover Y based on X and Z. In an ECC encryption system, point addition is always connected to the output of point multiplication. In this case, there is no need to transform to the affine coordinate system, since all point operations can be done in the projective coordinate system. The affine coordinate is only required when the key is transmitted, in which case only affine coordinate x is needed. Therefore the point multiplication unit should support the output of projective coordinate X, Y, Z, and affine coordinate x. As a result, the latency of the proposed architecture is comprised of three parts, the Montgomery Ladder, the coordinate transformation and one modular inversion to obtain x. The Montgomery Ladder takes 3 cycles per bit of input k, the total number of clock cycles is $3*163 = 489$ clock cycles. The coordinate transformation consists of 10 multiplications. With two parallel multipliers, we managed to reduce it to 6 clock cycles. Due to the special architecture of modular inversion to improve timing, calculating modular squaring as well as 2^2th powering needs 2 clock cycles respectively and 2^5th powering needs 3 clock cycles. Higher order powering operations are implemented by applying lower order powering units iteratively. Thus, latency for 2^{10}th powering, 2^{20}th powering, 2^{40}th powering and 2^{81}th powering are 4 clock cycles, 6 clock cycles, 10 clock cycles and 18 clock cycles. Therefore according to Algorithm 4, the total delay of modular division (modular inversion with an multiplication) is $4*2$ (squaring) + 2 (2^2th powering) + $3+4+6+10+18+1$ (multiplication) = 52 clock cycles. The total latency of point multiplication is $489 + 6 + 52 = 547$ clock cycles, which is the worst-case latency. When only projective coordinate is required, the modular inversion can be bypassed and 52 clock cycles can be reduced, leaving only 495 clock cycles required to finish the point multiplication operation.

V. IMPLEMENTATION RESULTS AND COMPARISONS

The proposed architecture is implemented on Xilinx Virtex-5 and Virtex-7 platform by Synplify 2018. Although the proposed architecture is feasible on all binary fields over Koblitz curve, only GF(2^{571}) and GF(2^{163}) are implemented to reflect universality since they are the two NIST-recommended GF(2^m) with the longest and the shortest bit length of m. The implementation results are shown in Tab. 2 and Tab. 3. The performances of different designs are evaluated by the product of implementation area (LUT number) and latency (number of clock cycles * period). Since our design is speed-oriented, latency and performance are key factors of the comparison.

The proposed design on GF(2^{163}) reaches the frequency of 211 MHz, and finishes one operation in 2.6 μs (547 clock cycles) at the cost of 29309 LUTs on Virtex-5. The same design reaches frequency of 320.5 MHz, and latency of 1.7 μs at the cost of 28911 LUTs on Virtex-7. Tab. 2 lists all existing designs on GF(2^{163}) in recent years, which aimed at realizing high-speed point multiplication operation. Among all published designs on Virtex-5 and Virtex-7, total latency of the proposed design is state-of-the-art. Meanwhile, the implementation achieves a relatively balanced performance, with high frequency, low latency and acceptable area cost. Consequently, it is especially suitable for high speed ECC applications.

Compared to [32], our design takes about 1/8 number of clock cycles under the frequency of 0.7 times of their design, achieving approximately 5.4 times speed, with comparable performance on Virtex-5. On Virtex-7, the frequency of the proposed design is improved to 0.8 times of their design, which leads to 6.2 times speed. The structure of two 55-bit modular multipliers proposed in [33] results in 40% less area than our proposed architecture on Virtex-5. As for latency, it takes six times clock cycles under 20% higher frequency, therefore it needs five times latency than the proposed design. The structure with one 163-bit multiplier was implemented in [12], [34], and [35] to finish the point multiplication operation. Since the multiplier dominates the area of the design, and their number of multipliers is half of ours, their designs cost fewer hardware resources. But their total latencies are 4.2, 3.65, and 3.3 times longer than the proposed design, as a

TABLE 2. FPGA implementation result on $GF(2^{163})$.

Works	$GF(2^m)$	FPGA Device	Number of Multipliers	Clock Cycle	Freq(MHz)	LUTs	Computation time(μs)	Performance
[3]	163	v5	1 163bit	3960	228	11400	17.36	198
[32]	163	v5	1 41bit	4168	296	3958	14.06	56
[33]	163	v5	2 55bit	3379	262	17305	12.9	223
[34]	163	v5	1 163bit	2189	199	18505	11	204
[12]	163	v5	1 163bit	1397	147	10195	9.5	97
[35]	163	v5	1 163bit	1429	167	10176	8.6	88
[20]	163	v5	3 81bit	1371	250	22936	5.48	126
[36]	163	v5	2 163bit	780	153	29095	5.1	148
[22]	163	v5	1 163bit	1119	228	16090	4.91	79
[21]	163	v5	1 163bit	1363	194	9470	4.6	44
[22]	163	v5	3 163bit	450	113	42192	3.99	168
Our work	**163**	**v5**	**2 163bit**	**547**	**211**	**29309**	**2.6**	**76**
[36]	163	v7	2 163bit	780	223	27105	3.5	95
[32]	163	v7	1 41bit	4168	397	4271	10.51	45
[22]	163	v7	1 163bit	1119	352	14202	3.18	45
[22]	163	v7	3 163bit	450	159	41090	2.83	116
Our work	**163**	**v7**	**2 163bit**	**547**	**320.5**	**28911**	**1.70**	**48**

TABLE 3. FPGA implementation result on $GF(2^{571})$.

Works	$GF(2^m)$	FPGA Device	Number of Multipliers	Clock Cycle	Freq(MHz)	LUTs	Computation time(μs)	Performance
[21]	571	v5	1 571bit	4044	106	57547	38.2	2,198
[21]	571	v5	1 571bit	4639	127	58665	36.5	2,141
[21]	571	v5	1 571bit	5805	143	57616	40.6	2,339
Our work	**571**	**v5**	**2 571bit**	**1813**	**186**	**284600**	**9.74**	**2,772**
[22]	571	v7	1 571bit	3780	111	141078	34.05	4,804
[32]	571	v7	1 143bit	14403	250	38547	57.61	2,221
Our work	**571**	**v7**	**2 571bit**	**1813**	**267**	**290001**	**6.79**	**1,969**

result of excessive number of clock cycles and long critical path. Compared to [20], our design is 26% higher in area, but the number of clock cycles is only 40%, leading to 2.1 times speed and 66% better performance.

The architecture proposed by [36] is most similar to our design, based on two parallel multipliers. Compared to them, we have two major novelty. First, the algorithm of two parallel multipliers is optimized from 4CC Montgomery Ladder to 3CC Montgomery Ladder, resulting in 30% less clock cycles. Second, the proposed design of balanced full-precision multiplier and mixed-pipeline architecture leads to only half of critical path of [36]. The total latency is only half of theirs and the performance is nearly two times on Virtex-5 and Virtex-7. Compared to [21], the proposed design takes up three times area and 61% less clock cycles, at the similar frequency. The performance is 42% worse than [21], while the computation speed is 77% higher. There are two architectures proposed in [22] and one aims at high performance while the other aims at low latency. Compared to HPECC in [22], the frequency is the same while the number of clock cycles half and area less than twice on Virtex-5, therefore the performance of our design is more balanced. Compared to LLECC [22], a three-multiplier design, the area of our design is 30% less, number of clock cycles is 25% more, but the frequency is twice on Virtex-5. In other words, our proposed design with only two multipliers is smaller and actually faster than the design with three parallel multipliers. On Virtex-7, the gap is even larger as the area of our design is improved to 42% less. The result shows that our design with only two multipliers is 53% faster than LLECC on Virtex-5 and 66% faster on Virtex-7. Conceivably, the proposed architecture is not only balanced in overall performance, but also reaches lower latency than all previous designs.

Since larger field gets better security, it is predictable that more and more applications such as network servers are going to turn to larger fields gradually. Therefore, it is crucial to keep the frequency of the architecture consistently high through different fields. However, most designs cannot maintain high frequency when it comes to larger fields, making them less practical in real-world applications than lab experiments. Tab. 3 shows the comparisons on GF(2^{571}) on Virtex-5 and Virtex-7. The proposed architecture can still reach high frequency on GF(2^{571}), and the number of computation clock cycles is 1813. On Virtex-5, it achieves 186 MHz, 284600 LUTs, and the latency of 9.74 μs; On Virtex-7, it achieves 267 MHz, 290001 LUTs, and the latency of 6.79 μs. Since two parallel multipliers are implemented, the area of our design is higher, but the speed and overall performance remain high.

Compared to [21], even the design with the best performance and fastest speed is 31.7% lower than our proposed design in frequency, and 156% higher in number of clock cycles on Virtex-5. Although its area is only 25%, our speed is more than four times theirs, and the frequency is higher. In terms of the results of both GF(2^{163}) and GF(2^{571}), [21] has excellent overall performance on 163-bit platform, with high frequency and low area, but when it comes to 571-bit platform, the frequency drops more than half. It is therefore proven that [21] is more suitable for smaller fields with less security. On the contrary, our implementation on 571-bit platform only suffers an acceptable frequency drop of 10% to 15%, indicating that the proposed design is less sensitive to the growth of field size. The result of another high-performance design [22], is only half on area, but twice on clock cycles and 40% on frequency. Its overall computation time (or latency) is five times of the proposed design and the performance is 144% better than the proposed design. In other words, we use twice the area to trade 5 times speed and end up with a much better performance. Furthermore, the proposed design also has the potential of being reconfigurable, for the balanced full-precision multiplier can support dual-field with minor alterations. Reconfigurable architecture can reuse the area resources in different field, and compensate for the extra area cost.

In general, on Virtex-5 and Virtex-7, the proposed design achieves state-of-the-art latency compared to all exiting design on GF(2^{571}) and GF(2^{163}). Since the proposed design contains two multipliers working in parallel to speed up point multiplication, the area is larger than designs with one multiplier or shared hardware resources [21], [32] and HPECC in [22]. However, extra area cost leads to a huge improvement in latency; meanwhile, the performance of the proposed design is among the top. Compared with other speed-oriented designs, the proposed design is the best from the points of total latency, performance as well as high-speed consistence in different fields. It is obviously concluded that the proposed speed-oriented design is significant for high-speed applications.

VI. CONCLUSION AND FUTURE WORK

This paper introduces a novel high-speed point multiplication architecture for ECC on extended binary fields. The architecture is comprised of two parallel balanced full-precision multipliers to greatly reduce operational latency. An improved algorithm of the Montgomery Ladder is proposed to best utilize the parallel multipliers. And the modular inversion architecture is studied and optimized to improve the performance of the whole architecture of point multiplication. To verify the universality of the design, it is implemented on the smallest and the largest NIST-recommended extended binary fields. The FPGA implementation results based on Xilinx Virtex-5 and Xilinx Virtex-7 show that the proposed speed-orietend design achieves the highest speed and lowest latency, which outperforms all of existing works. Since the multiplier and its segments work in different bit-length which refer to different fields, the proposed architecture can also be upgraded to a reconfigurable design to support multiple-field point multiplication in the future.

REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 218, H. C. Williams, Ed. Berlin, Germany: Springer, 1985, pp. 417–426.

[3] M. Rashid, M. Imran, A. R. Jafri, and T. F. Al-Somani, "Flexible architectures for cryptographic algorithms—A systematic literature review," *J. Circuits, Syst. Comput.*, vol. 28, no. 3, 2018, Art. no. 1930003.

[4] T. Gúneysu and C. Paar, "Ultra high performance ECC over NIST primes on commercial FPGAs," in *Proc. Sof Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2008, pp. 62–78.

[5] A. Salman, A. Ferozpuri, E. Homsirikamol, P. Yalla, J.-P. Kaps, and K. Gaj, "A scalable ECC processor implementation for high-speed and lightweight with side-channel countermeasures," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs*, Dec. 2017, pp. 1–8.

[6] B. Halak, S. S. Waizi, and A. Islam, "A survey of hardware implementations of elliptic curve cryptographic systems," IACR Cryptol. ePrint Arch., 2016. [Online]. Available: http://eprint.iacr.org/2016/712.pdf

[7] B. Rashidi. (2017). "A survey on hardware implementations of elliptic curve cryptosystems." [Online]. Available: https://arxiv.org/abs/1710.08336

[8] G. Zhou, H. Michalik, and L. Hinsenkamp, "Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1057–1066, 2010.

[9] M. Imran, M. Rashid, A. R. Jafri, 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 and M. Rashid, "Architectural review of polynomial bases finite field multipliers over GF(2^m)," in *Proc. Int. Conf. Commun.*, Mar. 2017, pp. 331–336.

[11] K. Safiullah, J. Khalid, and S. Y. Ali, "High-speed FPGA implementation of full-word montgomery multiplier for ECC applications," *Microprocessors Microsyst.*, vol. 62, pp. 91–101, Oct. 2018.

[12] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of elliptic curve scalar multiplier on LUT-based fpgas for area and speed," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 5, pp. 901–909, May 2013.

[13] S. Liu, L. Ju, X. Cai, Z. Jia, and Z. Zhang, "High performance FPGA implementation of elliptic curve cryptography over binary fields," in *Proc. IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Sep. 2015, pp. 148–155.

[14] N. Gura *et al.*, "An end-to-end systems approach to elliptic curve cryptography," in *Proc. Revised Papers Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Feb. 2002, pp. 349–365.

[15] 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.

[16] 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.

[17] B. Ansari and M. A. Hasan, "High-performance architecture of elliptic curve scalar multiplication," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[18] C. Shu, K. Gaj, and T. El-Ghazawi, "Low latency elliptic curve cryptography accelerators for NIST curves over binary fields," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, Dec. 2005, pp. 309–310.

[19] 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.

[20] 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.

[21] L. Li and S. Li, "High-performance pipelined architecture of elliptic curve scalar multiplication over GF(^{2}m)," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 4, pp. 1223–1232, Jun. 2016.

[22] Z. U. A. Khan and M. Benaissa, "High-speed and low-latency ECC processor implementation over GF(^{2}m) on FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 1, pp. 165–176, Jan. 2016.

[23] A. R. Jafri, M. N. ul Islam, M. Imran, and M. Rashid, "Towards an optimized architecture for unified binary huff curves," *J. Circuits, Syst. Comput.*, vol. 26, no. 11, Apr. 2017, Art. no. 1750178.

[24] B. Rashidi, "Efficient hardware implementations of point multiplication for binary edwards curves," *Int. J. Circuit Theory Appl.*, vol. 46, no. 8, pp. 1516–1533, Aug. 2018.

[25] A. Chatterjee and I. Sengupta, "High-speed unified elliptic curve cryptosystem on FPGAs using binary huff curves," in *Progress in VLSI Design and Test*. Springer, 2012, pp. 243–251.
[26] M. A. Hasan, A. H. Namin, and C. Negre, "Toeplitz matrix approach for binary field multiplication using quadrinomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 3, pp. 449–458, Mar. 2012.
[27] B. Rashidi, R. R. Farashahi, and S. M. Sayedi, "High-performance and high-speed implementation of polynomial basis itoh-tsujii inversion algorithm over GF(2^m)," *IET Inf. Secur.*, vol. 11, no. 2, pp. 66–77, Mar. 2017.
[28] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of the Itoh-Tsujii inversion algorithm for enhanced performance on k-LUT based FPGAs," in *Proc. Design, Autom. Test Europe Conf. Exhibit.*, Mar. 2011, pp. 1231–1236.
[29] R. S. Sinha, R. Chester, and M. Debdeep, "Generalized high speed Itoh-Tsujii multiplicative inversion architecture for FPGAs," *Integr. VLSI J.*, vol. 45, no. 3, pp. 307–315, Jun. 2012.
[30] L. Parrilla, A. Lloris, E. Castillo, and A. Garcá, "Minimum-clock-cycle Itoh-Tsujii algorithm hardware implementation for cryptography applications over GF(2^m) fields," *Electron. Lett.*, vol. 48, no. 18, pp. 1126–1128, 2012.
[31] J. Li, Z. Li, C. Xue, J. Zhang, W. Gao, and S. Cao, "A fast modular inversion FPGA implementation over GF(^2m) using modified x^{2n} unit," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
[32] Z.-U.-A. Khan and M. Benaissa, "Throughput/area-efficient ECC processor using montgomery point multiplication on FPGA," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 62, no. 11, pp. 1078–1082, Nov. 2016.
[33] R. Azarderakhsh and A. Reyhani-Masoleh, "Efficient FPGA implementations of point multiplication on binary Edwards and generalized Hessian curves using Gaussian normal basis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1453–1466, Aug. 2012.
[34] A. P. Fournaris, J. Zafeirakis, and O. Koufopavlou, "Designing and evaluating high speed elliptic curve point multipliers," in *Proc. 17th Euromicro Conf. Digit. Syst. Design*, Aug. 2014, pp. 169–174.
[35] C. Rebeiro, S. S. Roy, and D. Mukhopadhyay, "Pushing the limits of high-speed GF(^{2}m) elliptic curve scalar multiplication on FPGAs," *Br J Haematol*, vol. 149, no. 2, pp. 72–76, 2012.
[36] Z. U. A. Khan and M. Benaissa, "High speed ECC implementation on FPGA over GF(^{2}m)," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug. 2015, pp. 1–6.

JIAKUN LI received the bachelor's degree from the Beijing Institute of Technology, Beijing, China, in 2015, where she is currently pursuing the Ph.D. degree with the Institute of Microelectronics. Her research interests include digital circuit design, VLSI implementation of cryptography algorithms, and hardware design of cryptography basic operations.

SHUN'AN ZHONG is currently a Professor with the Beijing Institute of Technology. He is also the Director of the Beijing Engineering Research Center of Silicon-Based High-Speed System-on-Chip and the Institute of Microelectronics and has served successively as the Director of the Department of Undergraduate Academic Affairs and the Dean of the School of Information and Electronics all with the Beijing Institute of Technology. His current and successive academic part-time positions include: the member of information and electronics major education steering committee for colleges and universities of the Ministry of Education, the evaluation expert of the National Science and Technology Award of the Ministry of Industry and Information Technology, the member of undergraduate education branch of the Chinese Institute of Electronics, and the Managing Director of the *Journal of Semiconductors*.

ZHE LI received the bachelor's degree from the Beijing Institute of Technology, Beijing, China, in 2018. He is currently with Bitmain Technologies Ltd. His research interests include cryptography algorithms, high performance computing, and the design of digital VLSI circuits for cryptography applications and block chain technology.

SHAN CAO received the B.S. and Ph.D. degrees in microelectronics from Tsinghua University, in 2009 and 2015, respectively. She is currently an Associate Professor with the School of Communication and Information Engineering, Shanghai Institute for Advanced Communication and Data Science, Shanghai University, China. Her current research interests include embedded SoC system design and implementation for cryptography, wireless baseband circuit design, and hardware acceleration of convolutional neural networks.

JINGQI ZHANG received the B.Sc. degree in electronic engineering from the Beijing Institute of Technology, Beijing, China, in 2017, where he is currently pursuing the M.Sc. degree. His research interests include digital circuit design, VLSI, and cryptographic computations.

WEIJIANG WANG received the B.Eng. and Ph.D. degrees in communication and information system from the Beijing Institute of Technology, Beijing, China, in 1999 and 2004, respectively, where he joined the Faculty of the School of Information and Electronics, in 2004. His research interests include image processing and array signal processing, and design and hardware implementation for cryptography.