IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 29, 2020, accepted January 9, 2021, date of publication January 13, 2021, date of current version January 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051282

Innovative Dual-Binary-Field Architecture for Point Multiplication of Elliptic Curve Cryptography

JIAKUN LI, WEIJIANG WANG, JINGQI ZHANG, YIXUAN LUO, AND SHIWEI REN
School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
Corresponding author: Shiwei Ren (renshiwei@bit.edu.cn)

ABSTRACT High-performance Elliptic Curve Cryptography (ECC) implementation in encryption authentication severs has become a challenge due to the explosive growth of e-commerce's demand for speed and security. Point multiplication (PM) is the most common and complex operation in ECC which directly determines the performance of the whole system. This article proposes a 6CC-6CC (clock cycle) dual-field PM architecture and a 6CC-4CC dual-field PM architecture based on maximizing utilization of Karatsuba multipliers and re-ordering schedule strategy in PM. The Montgomery Ladder algorithm used in PM is modified to a 4CC algorithm for better resource utilization and parallel computation. To solve the frequency drop problem while working on large finite field, the PM architectures for high and low field are carefully studied to have universal critical path length and balanced performance. Both of the architectures are implemented over GF(2^{571}) and GF(2^{283}) on Xilinx Virtex-5 and Virtex-7 FPGAs (Field-Programmable Gate Array) for comparison. The 6CC-6CC architecture is shown to have the best performance on GF(2^{571}), which achieves one PM operation in 17.44 μs using 81549 LUTs (Look-Up-Table) with the frequency of 197.2 MHz on Virtex-5, and 12.55 μs using 80970 LUTs with the frequency of 274.1 MHz on Virtex-7. The 6CC-4CC architecture performs better on GF(2^{283}) with the shortest computation time. It takes only 3.21 μs to finish one PM operation on Virtex-5 and 2.22 μs on Virtex-7, which are faster than all the previous designs. The implementation results prove that the proposed architectures have state-of-the-art performance as well as higher versatility for ECC designs.

INDEX TERMS Elliptic curve cryptography (ECC), dual-field point multiplication, montgomery ladder, field-programmable gate array (FPGA) implementation.

I. INTRODUCTION

Elliptic Curve Cryptography (ECC) is a widely-used public-key cryptographic algorithm, which was proposed by Koblitz [1] and Miller [2] separately in 1985. The security of ECC relies on Elliptic Curve Discrete Logarithm Problem (ECDLP), which provides ECC with an equivalent level of security with a shorter key compared with the common public-key cryptographic algorithm, RSA [3]. Thus, ECC has the advantages of faster computing speed, lower bandwidth as well as lower storage requirement. Commonly, ECC is implemented over a prime field GF(p) or a binary field GF(2^m) [4]. Due to carry-free property, ECC operations over binary fields are much more suitable for hardware implementation. When implementing ECC on hardware, many designers turned to Field-Programmable Gate Arrays (FPGAs) because of its low cost, flexibility and short development period.

With the prosperity of digital economy, e-commerce has been widely spread, and the security of e-commerce has become an important premise to ensure its working. Therefore, fast and reliable encryption authentication servers are essential in the process of e-commerce transaction. FPGA-based cryptographic processors/accelerators for servers have become an inevitable trend. The security of ECC is related to the field size. As the development of super computers, the field size of ECC applications has to be increased to ensure safety. However, when existing high-performance ECC hardware designs are modified to larger fields to increase safety, there will be a significant drop in operating frequency, resulting in less competitive

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

TABLE 1. Complexity and latency analysis of different multipliers over $GF(2^m)$.

Works	Type	GF(2^m)	#XOR	#AND	Critical Path	Latency
[7]	Digit-Serial	409	$2mw - w^2/2 + 3w/2$	$(2m - w)w$	$T_A + (\lceil log_2 w \rceil + 2)T_X$	104
[40]	Digit-Serial	409	$69/20m^{log_4 6} - 1/4m^{log_4 2} - 11/5$	$m^{log_4 6}$	$T_A + (1 + 3log_4 m)T_X$	105
[41]	Digit-Serial	409	$wm + (2w - 1)$	wm	$T_A + (\lceil log_2 w \rceil + 2)T_X$	105
[8]	Bit-Serial	409	$2m$	$2m$ (NAND)	$T_N + 2T_{XN}$	409
[42]	Bit-Serial	409	$2m$	$2m$	$T_A + T_X$	409
[9]	Systolic	409	$m^2 - 1$	m^2	$T_A + T_X$	207
[43]	Systolic	409	$1.5m^2 + 0.5m$	m^{2b}	$T_{NA} + T_X$	411
[10]	Systolic	409	$m^2(m - 1)/16 + m + 26$	m^2	$2T_X$	18
[11]	Bit-Parallel	versatile	$m(m - 1) + (n - 1)m + (r - 1)m$	m^2	$T_A + (log_2(m/n) + n + 2r)T_X$	2
[12]	Bit-Parallel	versatile	$m(m - 1) + (n - 1)m + (r - 1)m$	m^2	$T_A + (log_2(m/n) + n + 2r)T_X$	2

[1] T_A and T_X denote the delay of 2-input AND and 2-input XOR logic gates.
[2] For digit-serial multipliers, each operand is divided into k digits where every digit contains w bits ($k = \lceil m/w \rceil$).

performance as opposed to the design purpose. Thus, implementing a high-performance as well as high-security ECC design on FPGA remains a challenge.

A. RELATED WORK

Point multiplication (PM) is the most common and complex operation in ECC, thus the majority of reported literatures focused on improving the operation of PM in order to increase the overall performance of ECC.

In order to optimize PM, some works explored new opportunities in modular multipliers over binary fields, as modular multiplication is executed during the whole process of PM. Depending on the organization of computation, there are bit-serial/ digit-serial, systolic multipliers, and bit-parallel multipliers [5], [6]. The complexity and latency (Clock Cycles, CCs) comparison are shown in Table 1.

1) BIT-SERIAL/DIGIT-SERIAL MULTIPLIER

In bit-serial/digit-serial multipliers, results are generated serially, which will be available after a large amount of clock cycles. For resources-constrained applications, bit-serial/digit-serial multipliers are preferred due to area-efficient property. In [7], a fully digit-serial polynomial basis GF(2^m) multiplier was proposed, where both the operands enter the architecture concurrently at digit-level. S. Pillutla proposed a modified interleaved modular reduction multiplication algorithm and its bit-serial sequential architecture in [8]. The multiplier achieves an improvement of 39% in area and 17% in area-delay product estimations for GF(2^{409}). It is obvious that bit-serial/digit-serial multipliers achieve area-efficient at the sacrifice of latency, which are rarely used in high-performance ECC designs.

2) SYSTOLIC MULTIPLIER

Systolic structures have the advantages of regularity, modularity, and concurrency, which offer high-throughput while the area and latency are usually large. Reference [9] proposed a systolic architecture for polynomial basis over GF(2^m) multiplier based on trinomial irreducible polynomials, which achieves 10% reduction in area complexity when compared with the best existing area-efficient multiplier over GF(2^{409}). Reference [10] analyzed the hidden systolic penalties in terms of register complexity and latency of computation in multipliers and proposed a systolic-like and a super-systolic-like structure for multiplication over GF(2^m).

3) BIT-PARALLEL MULTIPLIER

In bit-parallel multipliers, all input bits are handled in parallel. Almost all real-time applications turn to bit-parallel multipliers due to high speed and low latency. Practically, two or three pipeline stages are inserted to optimize the critical path. According to complexity, bit-parallel multipliers are divided into two categories, quadratic multipliers and subquadratic multipliers. Based on multipliers with quadratic complexity, [11] proposed a novel two-stage pipelined full-precision multiplier in high-performance (HP) PM architecture and a one-stage pipelined full-precision multiplier in low-latency (LL) architecture with careful scheduling for the Montgomery Ladder algorithm. Though [11] performs well on GF(2^{163}), the frequency decrease of [11] is also obvious on GF(2^{571}). Reference [12] proposed a novel speed-oriented architecture of PM based on a balanced full-precision multiplier that shows great consistency through different fields. The frequency of its implementation over GF(2^{571}) is only 13% lower than that over GF(2^{571}). However, the area cost (LUTs) is higher than most works. Karatsuba-Ofman multiplier (KOM) is a typical bit-parallel multiplier [13] which is able to reduce the complexity of a multiplication from $O(n^2)$ to $O(n^{log_2 3})$ at most [14]–[16]. Reference [17] analyzed hardware complexity of KOMs on FPGA and ASIC (Application Specific Integrated Circuits) and gave optimum iteration steps of KOMs for an arbitrary bit-depth. Reference [18] proposed a fast PM design with a two-stage pipelined KOM. Besides, the Montgomery Ladder algorithm is modified to share execution paths, so that the critical path contains few extra digital logic apart from the multiplier accumulator. The frequency of this design remains at a high level on small fields (GF(2^{163}), GF(2^{233}) and GF(2^{283})). However, on a larger field GF(2^{571}), the frequency drops more than 50% compared with that on GF(2^{163}). Reference [19] carried out theoretical analysis of the quadratic and subquadratic multipliers and drew a conclusion that the quadratic multiplier is a better option for high-speed ECC implementation when area is irrespective.

Apart from multipliers, some works focused on increasing parallelization of PM, so as to realize a high-performance ECC. Reference [20] applied a fix-base comb PM architecture to perform regular PM by pre-computation. Choosing window size as four, low-complexity (LC) and low-latency (LL) architectures have been proposed. Due to pre-computation, PM is executed in parallel. These architectures are designed for applications of Elliptic Curve Digital Signature Algorithm (ECDSA) that use fixed curve and base point, so that PM can be calculated in parallel after pre-computation. However, this technique is not applicable for all ECC applications. Some works [21]–[26] focused on rearranging the data flow of PM to satisfy complex data dependency with parallel structures. In these designs, additional registers are always needed to store intermediate values or to break critical paths.

Some works applied PM over special elliptic curves to improve the performance of ECC. Over Koblitz curves, special cases of Weierstrass Curve, Frobenius mappings can be used to accelerate PM [27]–[29]. Recently, Li and Li [30] proposed a high-performance PM architecture by applying pipelined mixed-form double-digit scalar converter with compact recoding over Koblitz curves. Frobenius mappings were applied to get a tremendous promotion in computation time. Moreover, in recent years, ECC over Binary Edwards curve (BEC) [31] and Binary Huff curve (BHC) [32] were given extensive attention due to high security. Based on algorithm-level unified addition laws, ECC over BEC and BHC has inherent preventive of Simple Power Analysis (SPA) [33]–[38] at the cost of higher computational complexity. Imran *et al.* [39] proposed a flexible hardware architecture that implements point multiplication algorithm for both elliptic curve cryptography (ECC) and Binary Huff Curves (BHC) and users can trade off between the algorithmic execution time and different reliability/security levels.

B. MAIN CONTRIBUTION

Most existing high-performance ECC designs are only able to operate over one pre-set binary field. However, for real-time Internet-based security equipment, PM is required to be compatible with several fields, as to meet different security levels and protocols. To cope with this, integrating several single-field PM architectures directly seems to be a simple way, but this will definitely lead to rather low rates of resource utilization, since when one PM is carried out, all other PM parts are idle.

Moreover, considering the critical path, though most of high-performance ECC designs support all NIST-recommended curves, the authors preferred to implement designs over small fields, GF(2^{163}) (for binary field) and GF(256) (for prime field). References [11], [18] and [44] exploited high-speed PM architectures on small binary field GF(2^{163}) with high operating frequency (more than 200 MHz). However, the high-frequency feature that these designs relied on to achieve low latency usually fails when it comes to larger fields like GF(2^{571}). This means that these high-performance architectures are not universal enough to apply in real-world applications.

To deal with these problems, based on the universal Weierstrass Curve, we proposed two novel PM architectures, 6CC-4CC (clock cycle) dual-field PM architecture and 6CC-6CC dual-field PM architecture, each of which is compatible with two binary fields and achieves equally competitive performance on both supported fields. In this case, our design can meet different security levels and protocols. The FPGA synthesis results show that the performance of our designs ranks top among related works.

The main contributions of this article are as follows:

1) Based on the Montgomery Ladder, a modified 4CC PM algorithm is proposed for high performance hardware implementations. When implementing two sets of modified 4CC PM algorithm on three modular multipliers, the advantages in frequency and efficiency of 6CC algorithm and the advantages in speed of 3CC algorithm can be gathered. It makes 4CC algorithm most suitable to be implemented in a speed-oriented PM design.
2) 6CC-6CC and 6CC-4CC dual-field PM architectures with different emphasis are proposed. The 6CC-6CC architecture has higher performance on the high field,[1] while the 6CC-4CC architecture works better on the low field.
3) The PM architectures for high and low field are carefully studied to have universal critical path length and similar frequency. This is the foundation of a high-performance dual-field design.
4) On either supported fields, the utilization of hardware resources is maximized to achieve equally competitive performances. The balance leads to state-of-the-art implementation results even when comparing with the best of other single-field architectures.

The rest of this article is organized as follows. Section II introduces related background and mathematic basis about ECC. Section III presents the proposed 6CC-6CC dual-field architecture, and the proposed 6CC-4CC dual-field architecture with modified 4CC PM algorithm is introduced in section IV. Section V presents the implementation results of the proposed architectures and comparisons with other works. Section VI concludes the paper.

II. PRELIMINARY

A. FINITE FIELD AND FIELD ARITHMETIC

Elliptic curves of ECC are usually defined over finite fields. Generally, there are two types of finite fields used by ECC, prime field GF(p) and binary field GF(2^m). The differences between these two fields are field arithmetic. Over binary field GF(2^m), the field arithmetic has a special property called ''carry-free'', which means that carries are ignored when the arithmetic is carried out. This property leads to tremendous

[1]In this article, high field stands for the larger field which the dual-field architecture supports, while low field stands for the smaller one.

convenience for hardware design. Thus, high speed ECC designs on FPGA/ASIC usually prefer binary field.

There are diverse representations for Elements over GF(2^m) and these representations result in different complexity of field arithmetic. Among representations, polynomial basis leads to simple multiplication, so it is commonly considered. By polynomial basis, elements over GF(2^m) can be represented as binary polynomials. Due to "carry-free", addition and subtraction are simplified to executing bitwise exclusive OR (XOR) over two binary polynomials' coefficients. In a sense, addition and subtraction are equivalent. Multiplication is carried out into two steps. The first is polynomial multiplication to get a $(2m-1)$-bit result. Then reduce this result by an irreducible polynomial $f(x)$ to obtain the final m-bit result. The National Institute of Standard and Technology (NIST) has recommended irreducible polynomials $f(x)$ for different binary fields [45]. Squaring is similar to multiplication; however, its first step is simplified to inserting zeroes into the coefficient sequence. And high order powering can be simplified in the same way. Inversion, which is the most complex arithmetic, is to find the multiplicative inverse x^{-1} of a given element x. Basically, there are two ways to compute inversion, extended Euclidean algorithm and Itoh Tsujii algorithm (ITA) [46].

B. MODULAR MULTIPLIER

Modular multiplication is executed in nearly every clock cycle from beginning to end during PM and multiplication is far more complex than addition and powering. Thus, a well-designed modular multiplier is crucial to achieve high-performance ECC. To exploit the trade-off between area and speed of multipliers in PM, there are mainly two types of multipliers, word-serial (digit-serial) multipliers and bit-parallel multipliers. For high-performance cases, large word-serial multipliers and bit-parallel multipliers are often used with pipeline stages inserted in order to shorten critical path delay.

Bit-parallel multipliers can be divided into quadratic multipliers and sub-quadratic multipliers. As for quadratic multipliers, [11] proposed a two-stage-pipeline full-precision multiplier, in which one of m-bit multiplier operand is divided into n number of w-bit operands. Then n sub-multipliers calculate the products of the other m-bit operand and all the w-bit operands simultaneously. The result of each sub-multiplier is $(m+w-1)$-bit long and all these results are stored in pipeline registers. n number of $(m+w-1)$-bit long results are aligned and XORed to figure out the $(2m-1)$-bit result, which is stored in the second pipeline stage. In the end, a fast modular reduction is done based on the chosen irreducible reduction polynomial.

The area complexity of quadratic multipliers is $O(n^2)$. To reduce the complexity, a special case of Toom-Cook multiplier, KOM is often used. KOM can reduce the area complexity down to $O(n^{log_2 3})$ at most. The strategy of KOM is shown in Fig. 1. As for a m-bit KOM (m is even), there are three $(m/2)$-bit sub-multipliers (when m is odd, it consists of one $\lfloor m/2 \rfloor$-bit sub-multiplier and two $\lceil m/2 \rceil$-bit sub-multipliers). The m-bit multiplication is divided into three steps, $a_h b_h$, $a_l b_l$ and $(a_h + a_l)(b_h + b_l)$.

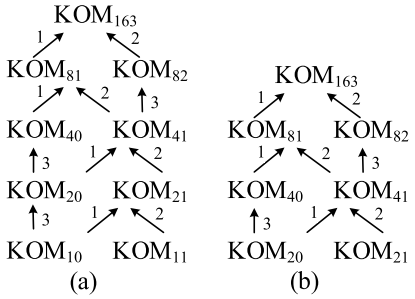

FIGURE 1. Area-delay efficient KOM for $m = 163$ on (a) four-input LUT FPGAs and (b) six-input LUT FPGAs.

$$\begin{aligned} A \cdot B &= (a_h x^n + a_l)(b_h x^n + b_l) \\ &= a_h b_h x^{2n} + (a_l b_h + a_h b_l) x^n + a_l b_l \\ &= a_h b_h x^{2n} + [(a_h + a_l)(b_h + b_l) \\ &\quad + a_h b_h + b_l] x^n + a_l b_l \\ \text{where } n &= \lfloor m/2 \rfloor, A = a_h x^n + a_l, B = b_h x^n + b_l. \end{aligned} \tag{1}$$

KOM can be applied in a recursive way. Large KOMs can be split into small KOMs. However, as the recursion goes on, the delay caused by calculating $[(a_h + a_l)(b_h + b_l) + a_h b_h + a_l b_l]$ will definitely affect critical path, which will lead to low operating frequency. Reference [17] shows the variation of KOMs' area and total delay when the recursive step changes and gives the best trade-off of KOM on different FPGAs in the end.

C. POINT MULTIPLICATION

A non-supersingular elliptic curve over GF(2^m) can be represented as:

$$y^2 + xy = x^3 + ax^2 + b. \tag{2}$$

Points over elliptic curve E and an infinite point form a communicative finite group based on point addition (PA) and point doubling (PD) [47]. The mathematical expression for PM is $Q = kP$, where P is point of the given curve and k is a positive integer. The result Q is also a point on this curve. For most common cases of point multiplication in which k and P can be flexibly configured, PM can be achieved by iterative scheduling of PA and PD based on each polynomial coefficient of k. And the specific operation of PA as well as PD can be carried out by executing several finite field arithmetic operations.

To resist Simple Power Analysis (SPA), the execution of PM has to be uniform when $k_i = 0$ (k_i stands for the i-th polynomial coefficient of k) and $k_i = 1$. The common strategy is to use the Montgomery Ladder. In affine coordinate, PA and PD, the operations executed in each iteration both include inversion over GF(2^m), which makes PM rather time-consuming. To cope with this problem, affine coordinate (x, y) is usually replaced with Lopez-Dahab (LD) projective

coordinate (X, Y, Z) [48]. As a result, inversions are only executed at the end of PM. Meanwhile, due to the constant difference between two intermediate variables in the iteration of the Montgomery Ladder, the y coordinates can be omitted during iteration and recovered at the end of the Montgomery Ladder. Thus, during PM, only X-coordinate and Z-coordinate are calculated iteratively, as shown in Algorithm 1.

Algorithm 1 Montgomery Ladder in the LD Coordinate System

Input: $k = (k_{t-1}, \cdots, k_1, k_0)_2$, $k_{t-1} = 1$, $P(x_P, y_P) \in GF(2^m)$)
Output: $Q = kP = (X_3, Y_3, Z_3)$
$(X_1, Z_1) \leftarrow (x_P, 1), (X_2, Z_2) \leftarrow ({x_P}^4 + b, {x_P}^2)$
for $i = t - 2$ to 0 **do**
  if $k_i = 1$ **then**
    $T \leftarrow Z_1, (X_1, Z_1) \leftarrow (x_P(X_1Z_2 + X_2Z_1)^2 + X_1X_2 \cdot T \cdot Z_2\ (X_1Z_2 + X_2Z_1)^2)$
    $T \leftarrow X_2, (X_2, Z_2) \leftarrow ({X_2}^4 + b{Z_2}^4\ T^2{Z_2}^2)$
  else
    $T \leftarrow Z_2, (X_2, Z_2) \leftarrow (x_P(X_1Z_2 + X_2Z_1)^2 + X_1X_2 \cdot T \cdot Z_1\ (X_1Z_2 + X_2Z_1)^2)$
    $T \leftarrow X_1, (X_1, Z_1) \leftarrow ({X_1}^4 + b{Z_1}^4\ T^2{Z_1}^2)$
  end if
end for
$X_3 \leftarrow X_1x_PZ_1Z_2$
$Y_3 \leftarrow (x_PZ_1 + X_1)[(X_1 + x_PZ_1)(X_2 + x_PZ_2) + ({x_P}^2 + y_P)Z_1Z_2]x_PZ_1^2Z_2 + y_P(x_PZ_1^2Z_2)^2$
$Z_3 \leftarrow x_pZ_1^2Z_2$
return $Q(X_3, Y_3, Z_3)$

According to Algorithm 1, the modular operations in each iteration are modular multiplications, modular squares and modular additions. Since the resource and time required in a modular multiplication are much more than in a modular addition or squaring, it is most important to organize modular multiplication well, in order to gain better efficiency. Therefore, the six modular multiplications in each iteration should be placed into six steps, and other modular operations should be inserted to the six separate steps to ensure the data dependency is sorted. This is called a 6-step Montgomery Ladder Algorithm, as shown in Algorithm 2.

III. 6CC-6CC DUAL-FIELD ARCHITECTURE

A. TRADITIONAL DUAL-FIELD PM AND ITS LIMITATIONS

Based on the 6-step Montgomery Ladder which is shown in Algorithm 2, a normal 6CC single-field architecture [11] (shown in Fig. 2) can be easily transformed to a dual-field one by updating the parts that are related to the field parameters. For example, for binary fields GF(2^m) and GF(2^n), the multiplication before reduction which is simple multiplication with addition replaced with exclusive-or, the statemachine which is determined by the algorithm, and the datapath (registers and MUXs in Fig. 2) can all be shared. Only modular reduction, modular squaring, and maybe other

Algorithm 2 6-Step Montgomery Ladder Algorithm When $k_i = 1$

Input: $R_1(X_1, Z_1), R_2(X_2, Z_2), k_{i+1}$
Output: $R_1(X_1, Z_1), R_2(X_2, Z_2)$
if $(k_{i+1} = 1)$ **then**
  step1: $Z_1 \leftarrow X_2Z_1; A \leftarrow Z_2$
  step2: $X_1 \leftarrow X_1Z_2; Z_2 \leftarrow A^2; R_2 \leftarrow A^4; A \leftarrow X_2$
else if $(k_{i+1} = 0)$ **then**
  step1: $X_1 \leftarrow X_1Z_2; A \leftarrow Z_2$
  step2: $Z_1 \leftarrow X_2Z_1; Z_2 \leftarrow A^2; R_2 \leftarrow A^4; A \leftarrow X_2$
end if
step3: $X_2 \leftarrow bR_2 + A^4; R_1 \leftarrow A^2$
step4: $Z_2 \leftarrow R_1Z_2; A \leftarrow X_1 + Z_1$
step5: $X_1 \leftarrow X_1Z_1; Z_1 \leftarrow A^2$
step6: $X_1 \leftarrow xZ_1 + X_1$
return $R_1(X_1, Z_1), R_2(X_2, Z_2)$

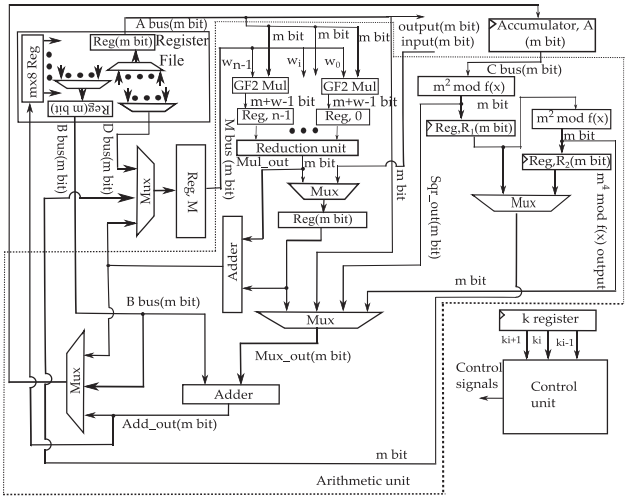

FIGURE 2. A normal 6CC single-field PM architecture from [11].

modular power units which are related to field parameters need to be added, together with some multiplexers attached to these newly-added units and the original ones to select the chosen field.

When the two chosen supported fields GF(2^m) and GF(2^n) have a certain relation such as $m \approx 2 * n$, the limitation of this dual-field architecture is revealed. When working on the low field GF(2^n), only the lower n bits of the datapath and the multiplier are used. In this case, a lot of the resources are wasted, resulting in low performance during low-field(GF(2^n)) operations. Such a design has lost its purpose of working efficiently on both fields.

A simple update to the traditional architecture is to utilize the unused logic by implementing another low-field 6CC operation on the higher n bits of the datapath and the multiplier. This method can evidently increase the utilization of the datapath to nearly 100% (since $m \approx 2 * n$), however the rate of rise in multiplier utilization depends on the design of the multiplier.

B. KARATSUBA MULTIPLIER

The multiplier implemented in the proposed architecture is a Karatsuba multiplier, as introduced in Sec. II-B. In the situation of $m \approx 2*n$, for an m-bit Karatsuba multiplier, three n-bit sub-multipliers are implemented on the top partition level, processing $a_1 * b_1, a_2 * b_2, (a_1 + a_2) * (b_1 + b_2)$ separately. Apparently, this multiplier has the potential of supporting only one modular multiplication on GF(2^m) but three modular multiplications on GF(2^n). If we apply two sets of low-field 6CC operations (each requiring one multiplication each cycle) as mentioned in the last subsection, two out of three of the sub-multipliers are used, increasing the utilization of multiplier to 66%. With the help of the Karatsuba multipliers, the improved 6CC-6CC dual-field architecture can calculate twice as fast on the low field, at the cost of only a few more small modular power units and modular reduction units which will not significantly influence the performance of high field.

C. IMPROVED 6CC-6CC DUAL-FIELD ARCHITECTURE

1) ARCHITECTURE

The detailed architecture of the proposed 6CC-6CC dual-field PM is shown in Fig. 3. It consists of one dual-field modular square unit, one dual-field modular quadruplicate unit, one Karatsuba multiplier, one $2w$ (w is introduced later)-bit datapath and two sets of statemachines. The structure of a dual-field modular square unit is shown in Fig. 4. Compared to the one used in traditional dual-field architectures, one more GF(2^n) modular square unit is used to support two GF(2^n) parallel calculations. The FSM in the architecture means actually two statemachines, one for higher w bit, and one for lower w bit. When working in high field, two statemachine jump exactly the same, using the same k. When working in low field, two statemachine jump on different ks and calculate in parallel.

Compared to the traditional dual-field architectures, this novel architecture utilized one more statemachine, one more modular quadruplicate unit on the low field, one more modular square unit on the low field and a few MUXs. The cost of area is little, but the gain in the low-field calculation speed is 100%.

2) RESOURCE SHARING

The Karatsuba multiplier and the datapath are shared as illustrated in Fig. 5. The w in Fig. 5 is the minimum value that is larger than $2 * n$ and m where $m \approx 2 * n$. To simplify the explanation, we take common fields $m = 571$ and $n = 283$ and $w = 286$ as an example.

During GF(2^{283}) mode, the lower 286 bits and the higher 286 bits of the datapath are used to transport two sets of data. When entering the multiplier, due to the top-level segment of Karatsuba, these two sets go right into mul1 (shown in Fig. 3) and mul2 which are independent to each other (the result of mul_share will be influenced, but it is not used in this mode). The outputs of the sub-multipliers are strobed to two

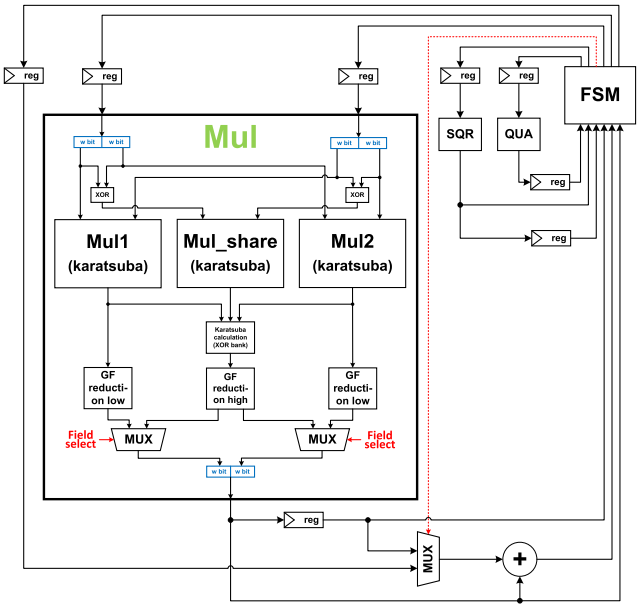

FIGURE 3. Detail of proposed 6CC-6CC dual-field architecture.

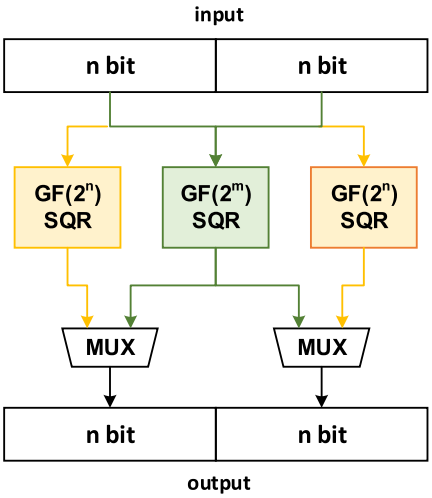

FIGURE 4. Structure of dual-field modular square unit.

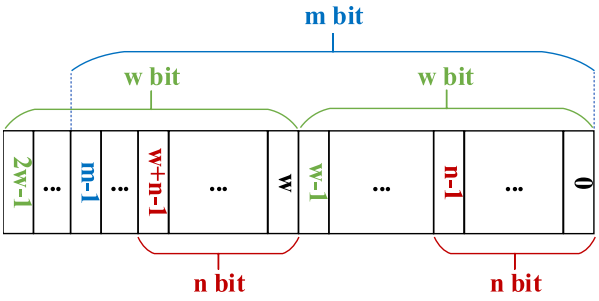

FIGURE 5. Illustration of bit distribution.

GF(2^{283}) modular reduction units and then two multiplexers to organize the results back to pattern shown in Fig. 5, before going back to the datapath again.

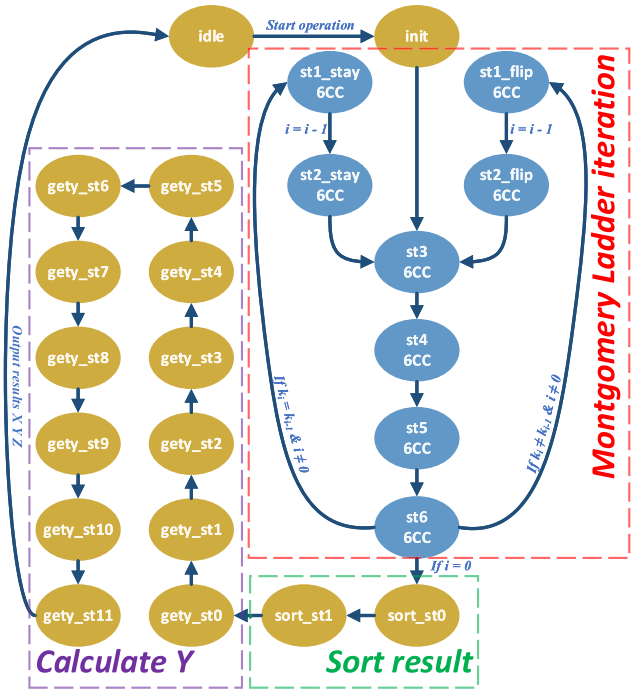

FIGURE 6. 6cc-6cc statemachine.

During GF(2^{571}) mode, the lower 286 bits and the higher 286 bits of the datapath are controlled by different statemachine that jump the same way, so they can be viewed as a whole. When entering the multiplier, the three sub-multipliers now work in the mechanism of Karatsuba to present only one result, which will then go through GF(2^{571}) modular reduction and into the datapath again.

3) STATEMACHINE AND DATA FLOW

The statemachine in the proposed 6CC-6CC architecture is shown in Fig. 6 When the operation starts, the state jumps to *init* from *idle*, to set initial values of X_1, X_2, Z_1, Z_2 as mentioned in algorithm. Then the statemachine jumps to the most common iteration, from *st*0_*stay*/*st*0_*flip* to *st*6 iteratively until all bits of k are processed. Then the state jumps to *sort*_*st*0 to put the results of X_1, X_2, Z_1, Z_2 into registers. In the next state *sort*_*st*1, the result of X_1 and X_2, Z_1 and Z_2 are exchanged if the last bit of k is 0, or retained if the last bit of k is 1. The statemachine then jumps to the stage to calculate Y. As shown in algorithm, the calculation of Y needs 12 multiplications. The 12 states from *gety*_*st*0 to *gety*_*st*11 process one multiplication in each to get the final results of X, Y and Z.

To show the scheduling of hardware resources in the most important states, the data flow of the 8 states in the Montgomery ladder iteration is demonstrated below.

Whether in high field or low field, the data flow of this architecture follows the 6CC algorithm, which is related to both k_i and k_{i+1}. The combination of k_i and k_{i+1} includes $k_i = 0$ 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 $X1$ and $X2$, as well as $Z1$ and $Z2$, only the situations of $k_i = 1$ while $k_{i+1} = 1$, and $k_i = 1$ while $k_{i+1} = 0$ are shown respectively, in Fig. 7 and Fig. 8.

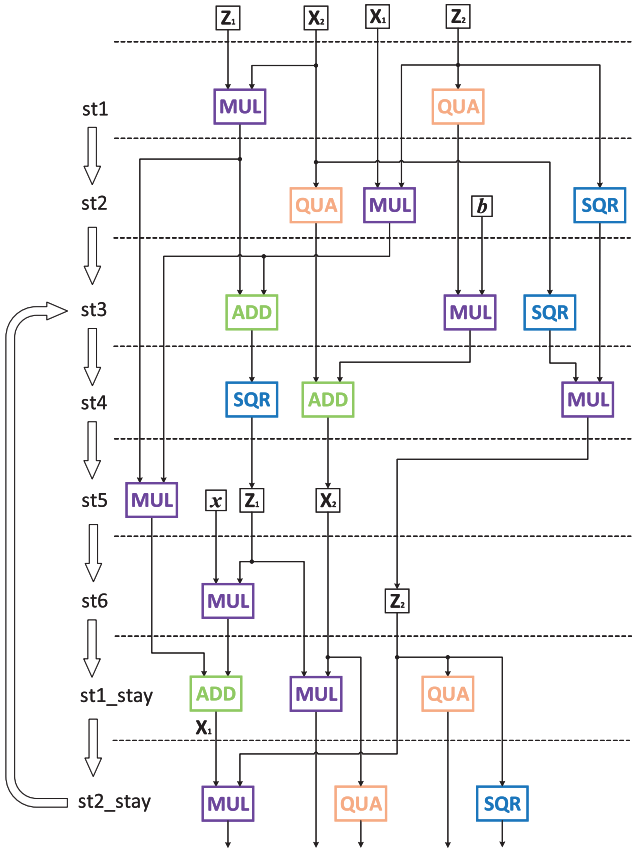

FIGURE 7. 6CC Data flow of $k_i = 1$ while $k_{i+1} = 1$.

In the figures, MUL stands for the modular multiplier, QUA stands for modular quadruple unit, SQR stands for module squaring unit, ADD stand for modular addition.

4) CRITICAL PATH

According to the dataflow shown in Fig. 7 and Fig. 8, the critical data dependency is that the result of the multiplier needs to be ready in 2 clock cycles. Therefore, apart from the input registers of the multiplier, one more stage of pipeline registers is inserted into the multiplier to shorten the critical path.

The critical path of 6CC-6CC architecture is shown in Fig. 9. Since the place of the pipeline registers inside the Karatsuba multiplier can be altered, first we remove it from consideration. Apparently, the path with most logic levels should be the one starts at the input of the modular multiplier, passes through the modular adder, and ends at the output registers of the FSM. Now add the pipeline registers into consideration. Since its place can be flexibly changed, it can be placed to divide the path mentioned above equally into two halves. Therefore, the critical path length of the 6CC-6CC architecture is: $(t_{KOM\_in\_GF(2^m)}+t_{MUX}+t_{add}+t_{MUXs\_in\_FSM})/2$. And the

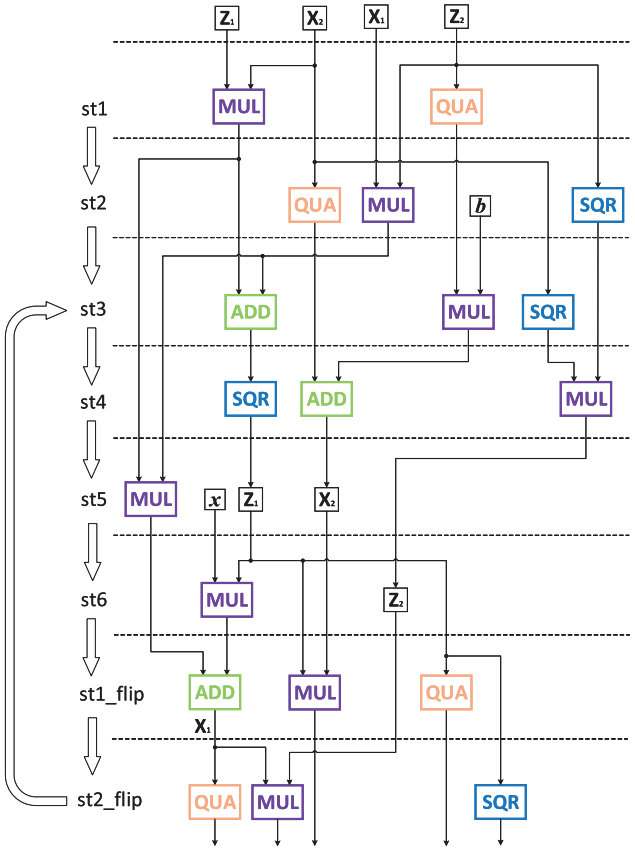

FIGURE 8. 6CC Data flow of $k_i = 1$ while $k_{i+1} = 0$.

critical path should be the path from multiplier input to the pipeline register inside multiplier, or the path from the pipeline register to the output register of the FSM. Compared with a single-field architecture, the critical delay is only $t_{MUX}/2$ higher.

5) LATENCY CALCULATION

According to the statemachine in Fig. 6, the Montgomery Ladder iteration takes 6 clock cycles to process one bit of k. After the iterations, two cycles are required to sort results and another 12 cycles are required to calculate y. Since the high and low mode utilize the same 6CC algorithm, the only difference is that in the low field two operations are done in parallel. Therefore, the latency calculations are:

high-field mode: $6 * m$(6CC)+2(get $X_1/Z_1/X_2/Z_2$) + 12(calculate y)

low-field mode: $6 * n$(6CC)+2(get $X_1/Z_1/X_2/Z_2$) + 12(calculate y) for two operations.

IV. 6CC-4CC DUAL-FIELD ARCHITECTURE

When working on low field, the proposed 6CC-6CC architecture only use two out of three sub-multipliers of Karatsuba. This leads to approximately 66% utilization of the multiplier. And since multiplier takes up most of the design's area, a large part of hardware resources is still wasted when working in low-field mode. In field applications, users usually use only one field at one time. If they choose to use the low field, the third sub-multiplier in the Karatsuba multiplier is completely useless. This awkward situation has prompted us to develop a new 6CC-4CC architecture that fully utilizes all the sub-multipliers in Karatsuba to further increase speed and hardware utilization in low field.

In order to fully use the sub-multipliers, the first step is to find a multi-multipliers architecture for the low field. Based on the 6-step algorithm introduced in Algorithm 2, the implementation can be done in 2CC with 3 multipliers, 3CC with 2 multipliers, 4CC with 2 multipliers, and 6CC with a single multiplier. 2CC introduces very complex cascaded multiplexers and data exchanges that result in large critical path [11], leaving only 3CC and 4CC to choose from.

Algorithm 3 4CC PM Algorithm

Input: $k, P(x_p, y_p)$
Output: $R_1(X_1, Z_1), R_2(X_2, Z_2)$
Initialization: $R_1(X_1, Z_1) \leftarrow (x_P, 1)$, $R_1(X_2, Z_2) \leftarrow ({x_P}^4 + b, {x_P}^2)$
for i from $t-2$ down to 0 **do**
  if $k_i = 1$ and $k_{i+1} = 1$ **then**
    St1: $Z_1 \leftarrow X_2 Z_1$
    St2: $X_1 \leftarrow Z_2 X_1$
    St3: $X_2 \leftarrow bZ_2^4 + X_2^4, Z_2 \leftarrow Z_2^2 X_2^2$
    St4: $X_1 \leftarrow X_1 Z_1 + x_p(X_1 + Z_1)^2, Z_1 \leftarrow (X_1 + Z_1)^2$
  else if $k_i = 1$ and $k_{i+1} = 0$ **then**
    St1: $X_1 \leftarrow Z_2 X_1$
    St2: $Z_1 \leftarrow X_2 Z_1$
    St3: $X_2 \leftarrow bZ_2^4 + X_2^4, Z_2 \leftarrow Z_2^2 X_2^2$
    St4: $X_1 \leftarrow X_1 Z_1 + x_p(X_1 + Z_1)^2, Z_1 \leftarrow (X_1 + Z_1)^2$
  else if $k_i = 0$ and $k_{i+1} = 0$ **then**
    St1: $Z_2 \leftarrow Z_2 X_1$
    St2: $X_2 \leftarrow X_2 Z_1$
    St3: $X_1 \leftarrow bZ_1^4 + X_1^4, Z_1 \leftarrow Z_1^2 X_1^2$
    St4: $X_2 \leftarrow X_2 Z_2 + x_p(X_2 + Z_2)^2, Z_2 \leftarrow (X_2 + Z_2)^2$
  else if $k_i = 0$ and $k_{i+1} = 1$ **then**
    St1: $X_2 \leftarrow X_2 Z_1$
    St2: $Z_2 \leftarrow Z_2 X_1$
    St3: $X_1 \leftarrow bZ_1^4 + X_1^4, Z_1 \leftarrow Z_1^2 X_1^2$
    St4: $X_2 \leftarrow X_2 Z_2 + x_p(X_2 + Z_2)^2, Z_2 \leftarrow (X_2 + Z_2)^2$
  end if
end for
return $R_1(X_1, Z_1), R_2(X_2, Z_2)$

A. 4CC PM ALGORITHM AND OTHERS

The 4CC PM algorithm is shown in Algorithm 3. In the first two cycles the 4CC algorithm calculates the first two multiplications in the 6-step PM algorithm, and it calculates two multiplications each cycle in the following two clock cycles. The difference of three competitive algorithms, 3CC, 4CC and 6CC are shown in Fig. 10. Accordingly, the first two states in 4CC is the same with 6CC, and the last two states in 4CC is the same with 3CC. 3CC combines the first two

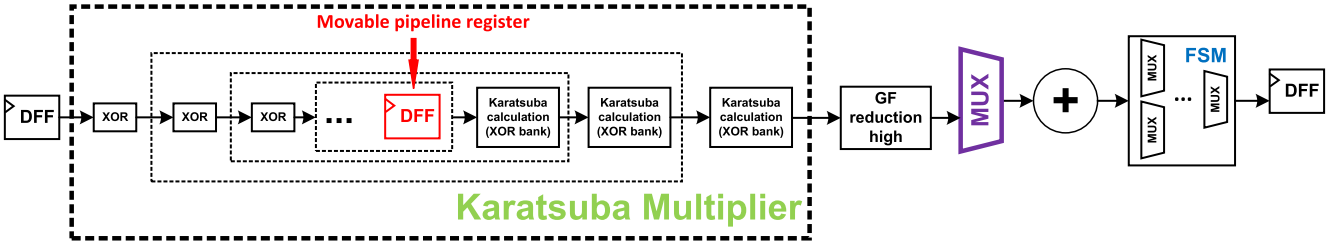

FIGURE 9. The critical path of 6CC-6CC architecture.

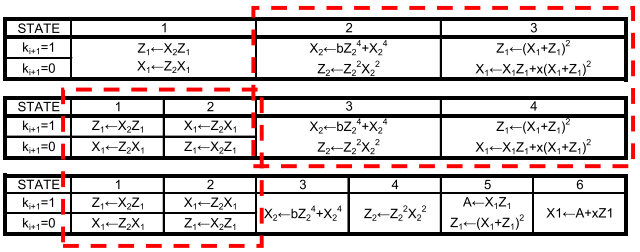

FIGURE 10. Comparison of 3CC 4CC and 6CC designs.

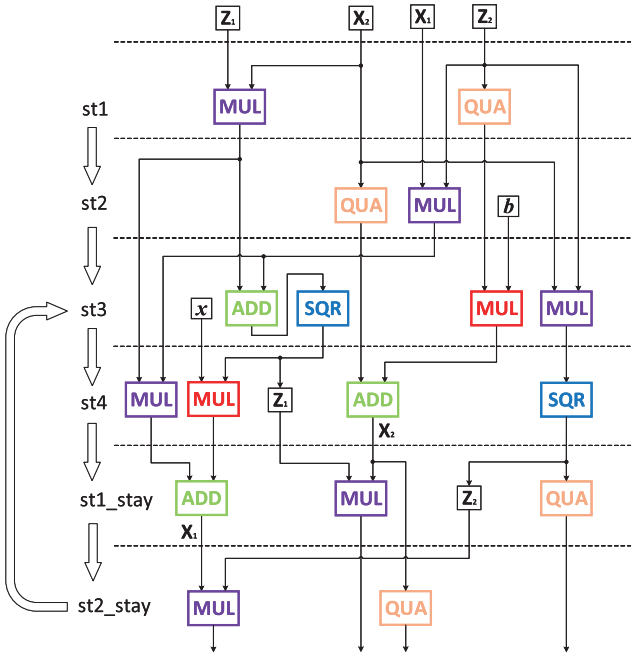

FIGURE 11. 4CC data flow of $k_i = 1$ while $k_{i+1} = 0$.

states in 4CC into one, leading to a critical data dependency between the last and the first state. 4CC combines step3 and step4 into one, and step5 and step6 into one, without introducing more critical data dependency than the original 6CC algorithm.

B. 4CC DATA FLOW

According to the Algorithm 3, the dataflow of 4CC PM can be easily concluded as shown in Fig. 11 and Fig. 12.

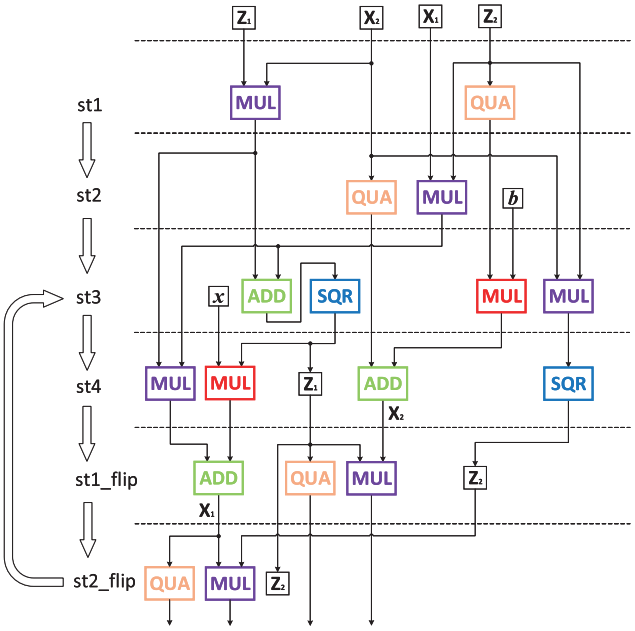

FIGURE 12. 4CC data flow of $k_i = 1$ while $k_{i+1} = 0$.

Again, only two scenarios out of four k_i and k_{i+1} combinations are shown, as the other two can be obtained by exchanging X_1 and X_2, Z_1 and Z_2 in Fig. 11 and Fig. 12.

C. CHOICE OF *CC ALGORITHM AND ITS INFLUENCE ON MULTIPLIER

In the past, 4CC is usually considered less efficient because the second multiplier only works half the time, as shown in Fig. 11 and Fig. 12. However, if we use 3 multipliers to do 2 sets of 4CC operations in parallel, the shared multiplier can work for the second main multiplier when it is free from the operations needed in the first main multiplier, as is shown in Fig. 13. Consequently, this proposed 4CC architecture can work as efficient as the 3CC architecture, while using all three sub-multipliers in the lower field.

From another standpoint, the pipeline stages inserted in the multiplier need to be considered when choosing the xCC architecture for the low field. In a dual-field architecture, the pipeline stages must be the same among both fields, otherwise the critical path will be determined by the one with less stages, influencing the performance of the field

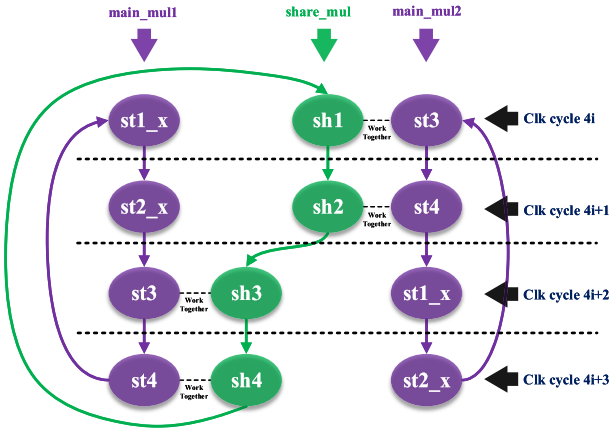

FIGURE 13. How the middle multiplier is shared to work with different main multipliers in 4CC algorithm.

with more pipeline stages. According to dataflow of 6CC in Sec. III-C3, dataflow of 4CC in Sec. IV-B and dataflow of 3CC from [12], the data dependency of 3CC can only allow one-stage pipeline to be inserted in the loop, while 4CC and 6CC can allow 2-stage pipeline in the loop and so have shorter critical path. In this case, 4CC and 6CC algorithm are more compatible to be unified in a dual-field architecture.

Combining these two points, we choose 4CC algorithm to implement in low field while remaining the 6CC design in high field.

D. 6CC-4CC ARCHITECTURE

Based on the methodology mentioned in the last section, we proposed the 6CC-4CC dual-field point multiplication architecture as shown in Fig. 15. The main calculation units include 4 modular adders, 4 modular square units and 2 modular quadruple units in low field, 1 modular square unit and 1 modular quadruple unit in high field and 1 dual-field Karatsuba modular multiplier.

1) DUAL-FIELD KARATSUBA MODULAR MULTIPLIER

As shown in Fig. 16, the dual-field Karatsuba modular multiplier is comprised of three separate Karatsuba sub-multipliers[2] with one-stage pipeline, three modular reduction units in low field, one modular reduction unit in high field, several multiplexers and XOR banks to sort the input and output according to the Karatsuba algorithm.

When working on the low field, the multiplexers of mul_share selects mul_sh as input. Three multipliers work separately, using three sets of inputs. The results go through three low-field modular reduction units to generate three outputs, mul_outl, mul_share_out and mult_out, which are used in two sets of 4CC PM calculations.

When working on the high field, the input multiplexers select the XOR result of lower and higher halves of mul_in1, and the XOR result of lower and higher halves of mul_in2. The results of these three sub_multipliers are strobed to the

[2]multipliers whose inputs are half the length of the main multiplier inputs.

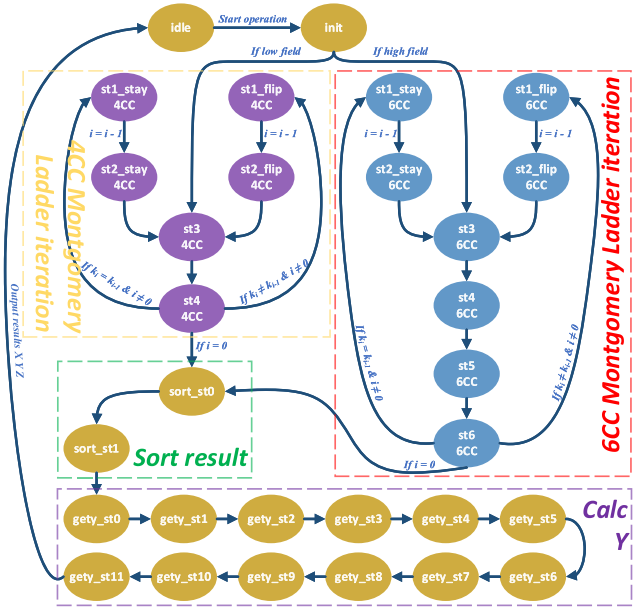

FIGURE 14. 6cc-4cc statemachine.

Karatsuba calculation XOR bank to generate the multiplication result based on Karatsuba algorithm, followed by the high-field modular reduction unit. After reduction, the modular multiplication result is re-divided in half to send back to the united datapath, following the same data structure of 6CC-6CC architecture shown in Fig. 3.

2) WORKING MECHANISM AND DATA FLOW

In this section, we will introduce how this proposed architecture utilizes its resources to support operations in both fields.

In low-field mode, the architecture can be viewed as two symmetric parts, each producing a 4CC calculation. Mul1 and the connected modular square units and adder, and the low-field modular quadruple unit in the left of the figure work in the whole 4 cycles, while mul_share works only in the first two cycles. The resources in the symmetric part of the mentioned units in Fig. 15 work with mul_share in the same way. To avoid conflict in using mul_share, the 4CC operation of the right side needs to start two cycles after the left side (also shown in Fig. 13), so that in the $4i$ and $4i+1$ cycles mul_share works for left, and in the $4i+2$ and $4i+3$ cycles mul_share works for right. Apart from this, two separate sides operate in the same 4CC dataflow discussed before in Sec. IV-B.

In high-field mode, the sub-multipliers work as a whole, following the same 6CC dataflow of 6CC-6CC architecture shown in Sec. III-C3.

3) STATEMACHINE

The statemachine of the share multiplier when working in the low field is shown in Fig. 13 and discussed in the last section. In the high field, the share multiplier and the main multipliers

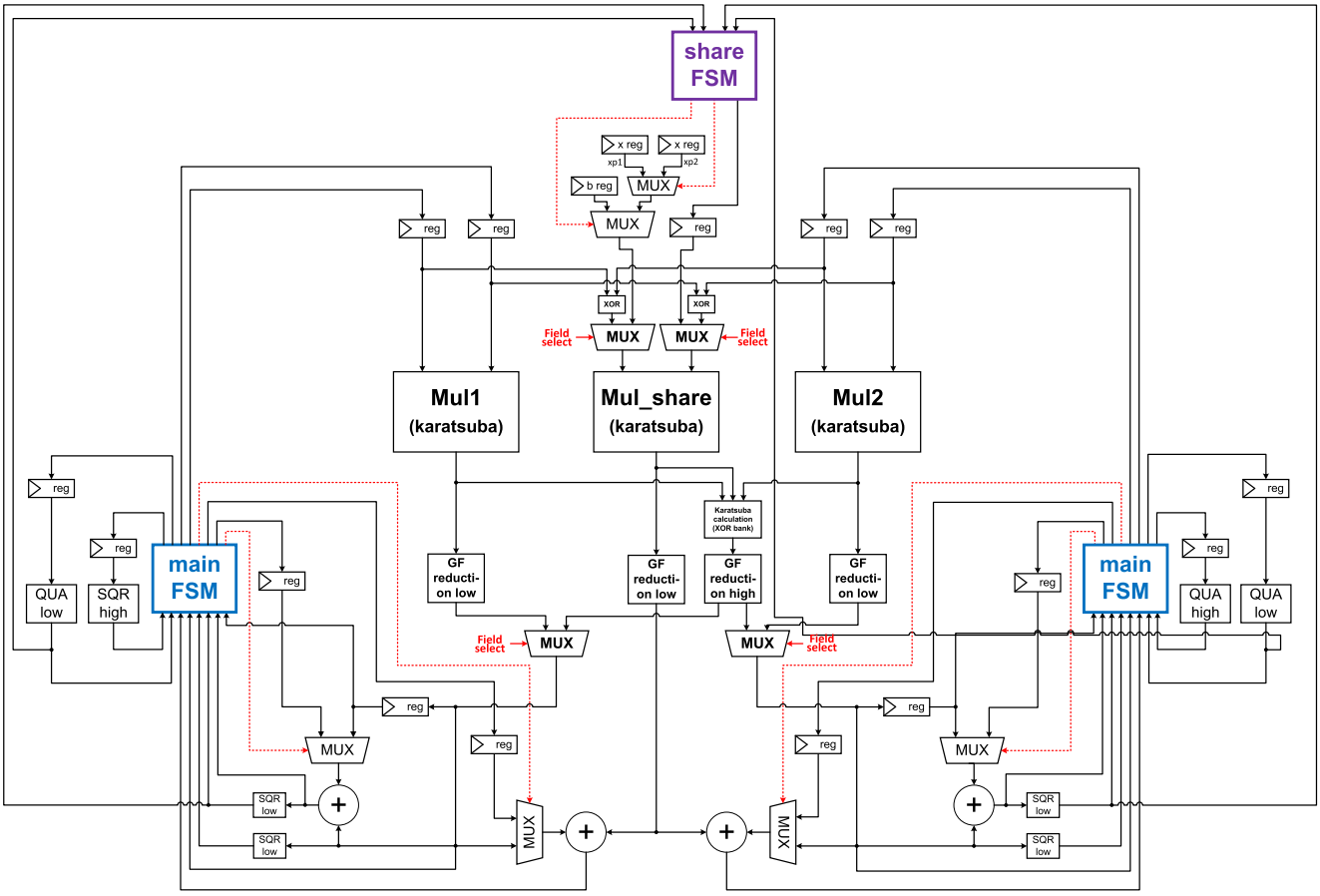

FIGURE 15. 6CC-4CC dual-field point multiplication architecture.

work as a whole in the Karatsuaba mechanism. Therefore, the main statemachine controls the two main multipliers in the low field, and the three multipliers as a whole in the high field. The main statemachine of the proposed 6CC-4CC architecture is shown in Fig. 14. The states to sort result, to calculate Y and the initial states are the same with the 6CC-6CC statemachine, while the Montgomery Ladder iterations include 4CC and 6CC. After *init* state, the state will jump to 4CC iteration if the architecture is working on the low field, or to 6CC iteration if the architecture is working on the high field. The detailed hardware scheduling of each state in 4CC and 6CC iteration are demonstrated in the corresponding data flow.

4) DIFFERENCE WITH 6CC-6CC

Critical path:

The critical path of 6CC-4CC architecture (shown in Fig. 17) is $(t_{karatsuba\_mul\_in\_GF(2m)}+t_{MUX}+t_{MUX}+t_{add}+t_{SQR}+t_{MUXs\_in\_FSM})/2$, only $(t_{SQR}+t_{MUX})/2$ higher than 6CC-6CC. The frequency difference is little.

Latency:

On high field GF(2^m), both architectures utilize the same 6CC dataflow, so the latency is the same $6*m$(6CC)+2(get $X_1/Z_1/X_2/Z_2$)+12(calculate y);

On low field GF(2^n), the latency of 6CC-4CC architecture is 2(gap for share_mul to support two ops)+$4*n$(4CC)+2(get $X_1/Z_1/X_2/Z_2$)+12(calculate y) for two operations. The average latency of one PM in 6CC-4CC is $2*n+8$, almost 50% lower than 6CC-6CC ($3*n+7$).

Area:

The gain in hardware resources is 3 modular adders, 3 modular square units and 1 modular quadruple units on the low field, and one set of statemachine for share_mul.

V. IMPLEMENTATION RESULTS AND COMPARISONS

A. OVERVIEW

To enable fair comparisons with existing works, the proposed 6CC-6CC and 6CC-4CC dual-field designs are implemented on Xilinx Virtex-5 series and Virtex-7 series FPGAs respectively using Synplify 2019 tool. During implementation, the field parameters m and n are instantiated with 571 and 283, which also satisfy the requirement $m \approx 2*n$. While GF(2^{163}), GF(2^{233}) and GF(2^{409}) are three other NIST-recommended that are equally important, the implementations of this article mainly focus on GF(2^{283}) and GF(2^{571}) for two reasons. First, GF(2^{571}) is the largest (most resource-consuming) among all NIST-recommended fields. It is one extreme scenario that most previous designs fail

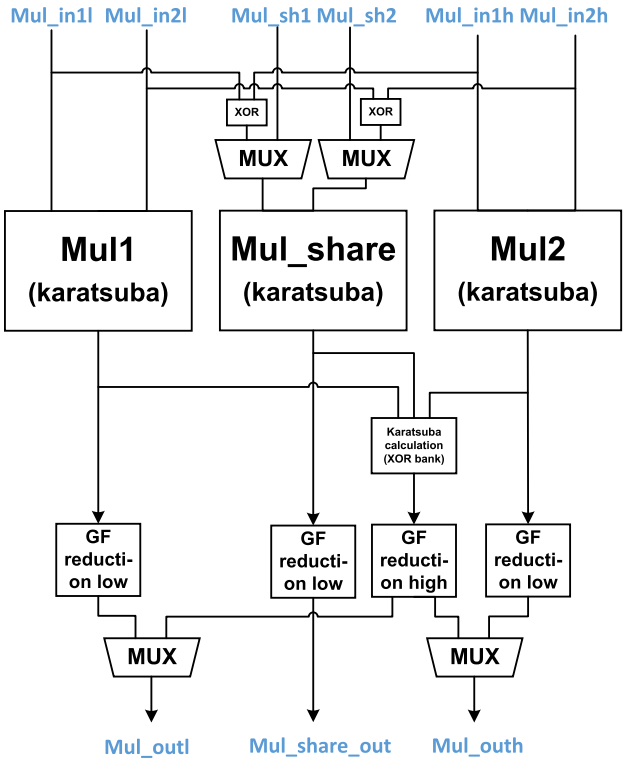

FIGURE 16. Three separate Karatsuba sub-multipliers in dual-field Karatsuba modular multiplier.

to retain the high performance. An equally balanced performance on GF2^{571}) with smaller fields can prove the universality and flexibility of the design. Second, the dual-field designs presented in the paper are intended to be extended to supporting all five binary fields. Supporting smaller fields based on a design on the largest field is easy, just add a few MUXs and field-related modular reduction units, the area and frequency will not be significantly influenced. On the contrary, supporting larger fields based on a design on smaller field may significantly influence the overall performance, since the datapath and multiplier which are most resource-consuming must be extended to support larger bit-length.

The synthesis results of the proposed designs compared with other works are shown in Table 2 and Table 3. Inside the tables, area which is quantified by LUTs (Look-Up-Table) and frequency is directly given by the synthesis tool, while the latency which is measured by Clock Cycles is derived from the calculation of hardware scheduling mentioned in Sec. III-C5 and Sec. IV-D4. Based on latency and the maximum frequency, total computation time of PM can be figured out. To better evaluate the classic trade-off between area and speed (shown in the form of computation time) existing in digital circuits, the parameter *Performance* is introduced. It is calculated by:

$$\begin{aligned} Performance &= \frac{\#Clock\ Cycles \times \#LUTs}{Freq.} \times 10^{-3} \\ &= Computation\ Time \times \#LUTs \times 10^{-3} \quad (3) \end{aligned}$$

One major difference between the proposed 6CC-6CC, 6CC-4CC architectures and other works is that each proposed architecture supports both GF(2^{283}) and GF(2^{571}). This means that the results of 6CC-6CC (or 6CC-4CC) architecture on both fields comes from the same implementation, hence have the same maximum frequency and occupied LUTs. For each of the other works that appears in both tables like [18] and [30], the results of GF(2^{283}) and GF(2^{571}) are based on the same architecture, but actually comes from different implementations. Since these different implementations can be optimized according to specific field parameters, the comparison between the proposed architectures and others is actually not fair. However, the performance of the proposed 6CC-6CC and 6CC-4CC designs still rank among the top.

B. THE SYNTHESIS RESULTS OF OUR WORKS

The proposed 6CC-6CC dual-field architecture reaches the maximum operating frequency of 197.2 MHz on Virtex-5, at the cost of 81549 LUTs. On Virtex-7, the maximum frequency of 6CC-6CC architecture is 274.1 MHz, and 80970 LUTs are occupied. For 6CC-6CC architecture, it takes 856 CCs to finish one PM on GF(2^{283}) while 3440 CCs on GF(2^{571}); thus, computation time is 4.34 μs (Virtex-5) and 3.12 μs (Virtex-7) on GF(2^{283}) and 17.44 μs (Virtex-5) and 12.55 μs (Virtex-7) on GF(2^{571}), respectively.

The proposed 6CC-4CC dual-field architecture reaches the maximum operating frequency of 178.7 MHz on Virtex-5, at the cost of 87067 LUTs. On Virtex-7, the maximum frequency of 6CC-4CC architecture is 258.6 MHz, and 86228 LUTs are occupied. As for timing, compared with 6CC-6CC architecture, the critical path of 6CC-4CC architecture is 10.4% longer than 6CC-6CC architecture on Virtex-5, and 6% longer on Virtex-7. These low percentages of timing differences are consistent with the theoretical analysis in Sec. IV-D4. As for area, 6CC-6CC architecture has subtle advantage in LUTs compared with 6CC-4CC architecture, 6.3% on Virtex-5 and 6.4% on Virtex-7. For 6CC-4CC architecture, it takes 573.5 CCs (on average) to finish one PM on GF(2^{283}) while 3440 CCs on GF(2^{571}). Based on frequency and latency (Clock cycles), computation time is 3.21 μs (Virtex-5) and 2.22 μs (Virtex-7) on GF(2^{283}) and 19.25 μs (Virtex-5) and 13.30 μs (Virtex-7) on GF(2^{571}).

C. COMPARISON ON GF(2^{571})

On GF(2^{571}), the 6CC-6CC architecture reaches the highest frequency among all existing works at the cost of reasonable LUTs. The performance of 6CC-6CC architecture is the best, and the performance of 6CC-4CC architecture is also among the top. The 6CC-6CC architecture outperforms 6CC-4CC architecture in every way on Virtex-5 and Virtex-7. The loss in performance on GF(2^{571}) of 6CC-4CC architecture is to give way to the improvement in performance on GF(2^{283}).

Among related works, [44] is the only other dual-field design implemented on GF(2^{283}) and GF(2^{571}). And both our proposed architectures are better than [44] on GF(2^{283})

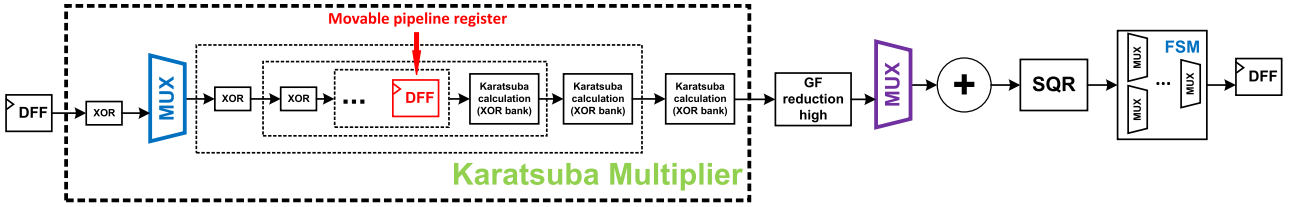

FIGURE 17. The critical path of 6CC-4CC architecture.

TABLE 2. FPGA implementation results on GF(2^{571}).

Works	GF(2^m)	FPGA Device	Number of Multipliers	Clock Cycle	Freq(MHz)	LUTs	Computation time(μs)	Performance
[44]	571	v5	1 191-bit	16362	162	29708	101.00	3001
[12]	571	v5	2 571-bit	1803	186	284600	9.74	2772
[18]	571	v5	1 571-bit	5805	143	57616	40.60	2339
[18]	571	v5	1 571-bit	4639	127	58665	36.50	2141
[30]	571	v5	1 571-bit	-	-	81164	18.51	1502
6cc_4cc	**571**	**v5**	**1 571-bit**	**3440**	**178.7**	**87067**	**19.25**	**1676**
6cc_6cc	**571**	**v5**	**1 571-bit**	**3440**	**197.2**	**81549**	**17.44**	**1423**
[11]	571	v7	1 571-bit	3780	111	141078	34.05	4804
[50]	571	v7	1 143-bit	14403	250	38547	57.61	2221
[12]	571	v7	2 571-bit	1813	267	290001	6.79	1969
6cc_4cc	**571**	**v7**	**1 571-bit**	**3440**	**258.6**	**86472**	**13.30**	**1150**
6cc_6cc	**571**	**v7**	**1 571-bit**	**3440**	**274.1**	**80970**	**12.55**	**1016**

TABLE 3. FPGA implementation results on GF(2^{283}).

Works	GF(2^m)	FPGA Device	Number of Multipliers	Clock Cycle	Freq(MHz)	LUTs	Computation time(μs)	Performance
[44]	283	v5	1 191-bit	8100	162	29708	50.00	1485
[26]	283	v5	3 47-bit	6720	189	26460	35.56	940
[26]	283	v5	3 47-bit	6347	189	25030	33.60	841
[18]	283	v5	1 283-bit	2329	213	20256	10.90	221
[20]	283	v5	1 283-bit	552	128	49313	4.31	212
[30]	283	v5	1 283-bit	-	-	30952	5.81	180
[51]	283	v5	1 283-bit	2339	320	24460	7.31	178
6cc_4cc	**283**	**v5**	**3 283-bit**	**573.5**	**178.7**	**87067**	**3.21**	**279**
6cc_6cc	**283**	**v5**	**2 283-bit**	**856**	**197.2**	**81549**	**4.34**	**354**
[20]	283	v7	1 283-bit	553	185	45932	2.99	137
[51]	283	v7	1 283-bit	23240	496	23240	4.72	109
[49]	283	v7	1 283-bit	-	337	24460	20.32	410
6cc_4cc	**283**	**v7**	**3 283-bit**	**573.5**	**258.6**	**86472**	**2.22**	**192**
6cc_6cc	**283**	**v7**	**2 283-bit**	**856**	**274.1**	**80970**	**3.12**	**253**

and GF(2^{571}) in the field of computation time as well as performance.

On Virtex-7, both proposed architectures are state-of-the-art designs, which perform 93.8% and 71.2% better than the best existing work [12]. On Virtex-5, the performance of 6CC-6CC architecture is 5.5% better than the best existing work [30]. The superiority in performance is evident, not to mention the ability to support more than one field. The performance of 6CC-4CC architecture on Virtex-5 is far better than [18] and [12], but is 10.4% worse than [30]. However, the high-performance PM architecture in [30] is designed to work only on single preset field over Koblitz curves, which is a special case of Simplified Weierstrass Curves. But the proposed 6CC-4CC design is compatible with all Simplified Weierstrass Curves cases, and supports both GF(2^{283}) and GF(2^{571}) fields. The advantages on flexibility compensate for the inferiority in performance, making the proposed 6CC-4CC architecture equally competitive with [30].

D. COMPARISON ON GF(2^{283})

Since our designs are dual-field, extra control logic is inevitably introduced into datapath, which will definitely lead to rise in occupied LUTs, increased length in critical path and thus decrease in performance. This effect is even more severe when the proposed architectures work on GF(2^{283}). In this case, the performance of the proposed designs may not be the best, but the speed increase brought by parallelization can still stand out in comparison with other works on GF(2^{283}).

On GF(2^{283}), the proposed 6CC-4CC dual-field architecture performs much better than 6CC-6CC architecture on Virtex-5 and Virtex-7, due to the great improvement in the utilization of hardware resources which is based on the modified 4CC PM algorithm. Therefore, we will take 6CC-4CC architecture as the example to compare with other works. The performance of 6CC-4CC architecture is better than [44] (432% on Virtex-5), [26] (201% on Virtex-5) and [49] (113.5% on Virtex-7).

Compared with [18], the performance of our 6CC-4CC architecture is 26.2% worse on Virtex-5. However, our design stands out in the aspect of computation time, which is 239% faster. Also, the implementation results in [18] shows that it only works well on small finite fields such as GF(2^{163}), GF(2^{233}) and GF(2^{283}). On large field like GF(2^{571}), the performance is much weaker due to the drop in frequency. If the architecture of [18] is extended to a dual-field design, the frequency of working on GF(2^{283}) will be dragged down by the less-competitive frequency of GF(2^{571}), not to mention the additional frequency loss brought by additional MUXs. This will make its performance worse than ours on both GF(2^{283}) and GF(2^{571}).

The performance of the proposed 6CC-4CC architecture is inferior to [20] (−31.6% on Virtex-5 and −40.1% on Virtex-7) and [51] (-56.7% on Virtex-5 and -76.1% on Virtex-7). These two works are designed for ECDSA, and the elliptic curve as well as the base point of their design in PM is fixed while the scalar changes. But in our design, both base point and scalar are unknown and they are both the input of our architecture. Thus, the application scope of our design is far broader. Besides, the computation time of our design is merely 3.21μs on Virtex-5 and 2.22μs on Virtex-7, indicating our design is more competitive when it comes to speed-oriented cases.

The performance of [30] is 55% better than our design on Virtex-5. However, as mentioned in Sec. V-C, [30] is only suitable for Koblitz curves on a single field whereas our design has better applicability.

The implementation results on GF(2^{283}) and GF(2^{571}) show that, the proposed 6CC-6CC architecture and 6CC-4CC architecture each has its prominence on performance, speed and applicability compared to other related works. These proposed architectures also stand out in the aspect of universality, since only a few works in literature have studied dual-field architectures, which are more related to actual field applications. Among the two proposed designs, 6CC-6CC is better at working on GF(2^{571}) than on GF(2^{283}), making it more suitable for applications that mainly works on GF(2^{571}); while the 6CC-4CC architecture has more balanced performance on both supported fields, and is better for applications whose working field is uncertain at the design period.

VI. CONCLUSION

In this article, we modified the Montgomery Ladder to 4CC PM algorithm for hardware and proposed a 6CC-6CC dual-field PM architecture and a 6CC-4CC dual-field PM architecture based on maximizing utilization of Karatsuba multipliers and re-ordering schedule strategy in PM. For PM over GF(2^n) and GF(2^m) which satisfy the relation, $m \approx 2 * n$, the proposed architectures can work as efficient. The universality of frequency on different fields is also excellent, rendering these proposed architectures high performance on all supported fields. The FPGA implementations on GF(2^{571}) and GF(2^{283}), Virtex-5 and Virtex-7 prove that our designs are state-of-the-art even when comparing with the top single-field architectures. In the future, the proposed architectures can also be easily extended to supporting all five NIST-recommended fields, by doing some minor updates on the modular squaring and reduction units.

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*, vol. 218, no. 1. Berlin, Germany: Springer, 1986, pp. 417–426.
[3] F. Rodríguez-Henríquez, N. A. Saqib, A. D. Perez, and C. K. Koc, *Cryptographic Algorithms on Reconfigurable Hardware*. Berlin, Germany: Springer, 2007.
[4] 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, Mar. 2019, Art. no. 1930003.
[5] M. Imran and M. Rashid, "Architectural review of polynomial bases finite field multipliers over GF(2^m)," in *Proc. Int. Conf. Commun., Comput. Digit. Syst. (C-CODE)*, Mar. 2017, pp. 331–336.
[6] S. Khan, K. Javeed, and Y. A. Shah, "High-speed FPGA implementation of full-word montgomery multiplier for ECC applications," *Microprocessors Microsyst.*, vol. 62, pp. 91–101, Oct. 2018.
[7] S. R. Pillutla and L. Boppana, "A high-throughput fully digit-serial polynomial basis finite field GF(2^m) multiplier for IoT applications," in *Proc. IEEE Region Conf. (TENCON)*, Oct. 2019, pp. 920–924.
[8] S. R. Pillutla and L. Boppana, "An area-efficient bit-serial sequential polynomial basis finite field GF(2^m) multiplier," *AEU-Int. J. Electron. Commun.*, vol. 114, Feb. 2020, Art. no. 153017.
[9] S. R. Pillutla and L. Boppana, "Area-efficient low-latency polynomial basis finite field GF(2^m) systolic multiplier for a class of trinomials," *Microelectron. J.*, vol. 97, Mar. 2020, Art. no. 104709.
[10] B. K. Meher and P. K. Meher, "Analysis of systolic penalties and design of efficient digit-level systolic-like multiplier for binary extension fields," *Circuits, Syst., Signal Process.*, vol. 38, no. 2, pp. 774–790, Feb. 2019.
[11] 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. (VLSI) Syst.*, vol. 25, no. 1, pp. 165–176, Jun. 2016.
[12] J. Li, S. Zhong, Z. Li, S. Cao, J. Zhang, and W. Wang, "Speed-oriented architecture for binary field point multiplication on elliptic curves," *IEEE Access*, vol. 7, pp. 32048–32060, 2019.
[13] A. Karatsuba, "Multiplication of multidigit numbers on automata," *Doklady Akad Nauk Sssr*, vol. 145, p. 595, Jan. 1963.
[14] 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.
[15] 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 13th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Sep. 2014, pp. 148–155.
[16] N. Gura, S. C. Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, "An end-to-end systems approach to elliptic curve cryptography," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2002, pp. 349–365.
[17] 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, Jul. 2010.
[18] 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, Aug. 2016.
[19] 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.
[20] R. Salarifard, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "A low-latency and low-complexity point-multiplication in ECC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 9, pp. 2869–2877, Sep. 2018.
[21] 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, 2013.

[22] 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.
[23] 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.
[24] C. Shu, K. Gaj, and T. Elghazawi, "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.
[25] 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, Aug. 2006.
[26] G. D. Sutter, J. Deschamps, and J. L. Imana, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Trans. Ind. Electron.*, vol. 60, no. 1, pp. 217–225, Jan. 2013.
[27] S. Okada, N. Torii, K. Itoh, and M. Takenaka, "Implementation of elliptic curve cryptographic coprocessor over GF(2^m) on an FPGA," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2000, pp. 25–40.
[28] K. Jarvinen and J. Skytta, "On parallelization of high-speed processors for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 9, pp. 1162–1175, Sep. 2008.
[29] R. Azarderakhsh and A. Reyhani-Masoleh, "High-performance implementation of point multiplication on Koblitz curves," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 1, pp. 41–45, Jan. 2013.
[30] 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.
[31] D. J. Bernstein, T. Lange, and R. R. Farashahi, "Binary edwards curves," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2008, pp. 244–265.
[32] J. Devigne and M. Joye, "Binary huff curves," in *Proc. Cryptographers Track RSA Conf.* Berlin, Germany: Springer, 2011, pp. 340–355.
[33] A. R. Jafri, M. N. U. Islam, M. Imran, and M. Rashid, "Towards an optimized architecture for unified binary huff curves," *J. Circuits, Syst. Comput.*, vol. 26, no. 11, Nov. 2017, Art. no. 1750178.
[34] 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.
[35] R. Azarderakhsh and A. Reyhani-Masoleh, "Parallel and high-speed computations of elliptic curve cryptography using hybrid-double multipliers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1668–1677, Jun. 2015.
[36] A. Chatterjee and I. Sengupta, "High-speed unified elliptic curve cryptosystem on FPGAs using binary huff curves," in *Proc. Prog. VLSI Design Test*. Berlin, Germany: Springer, 2012, pp. 243–251.
[37] 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.
[38] M. Rashid, M. Imran, A. R. Jafri, and Z. Mehmood, "A 4-stage pipelined architecture for point multiplication of binary huff curves," *J. Circuits, Syst. Comput.*, vol. 29, no. 11, Sep. 2020, Art. no. 2050179.
[39] M. Imran, M. Rashid, A. R. Jafri, and M. N. U. Islam, "Acryp-proc: Flexible asymmetric crypto processor for point multiplication," *IEEE Access*, vol. 6, p. 22 778–22 793, 2018.
[40] C.-Y. Lee, C.-S. Yang, B. K. Meher, P. K. Meher, and J.-S. Pan, "Low-complexity digit-serial and scalable SPB/GPB multipliers over large binary extension fields using (b,2)-way karatsuba decomposition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 11, pp. 3115–3124, Nov. 2014.
[41] S. H. Namin, H. Wu, and M. Ahmadi, "Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 441–449, Feb. 2017.
[42] S. E. Mathe and L. Boppana, "Design and implementation of a sequential polynomial basis multiplier over GF(2^m)," *Ksii Trans. Internet Inf. Syst.*, vol. 11, no. 5, pp. 2680–2700, 2017.
[43] S. Bayat-Sarmadi and M. Farmani, "High-throughput low-complexity systolic montgomery multiplication over GF(2^m) based on trinomials," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 4, pp. 377–381, Jan. 2015.
[44] K. C. C. Loi, S. An, and S.-B. Ko, "FPGA implementation of low latency scalable elliptic curve cryptosystem processor in GF(2^m)," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 822–825.
[45] N. Nist, "Recommended elliptic curves for federal government use," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. SP 800-186, 1999. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-186/draft
[46] 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.
[47] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin, Germany: Springer, 2006.
[48] J. López and R. Dahab, "Fast multiplication on elliptic curves over GF(2^m) without precomputation," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 1999, pp. 316–327.
[49] M. Imran, M. Rashid, A. R. Jafri, and M. Kashif, "Throughput/area optimised pipelined architecture for elliptic curve crypto processor," *IET Comput. Digit. Techn.*, vol. 13, no. 5, pp. 361–368, Sep. 2019.
[50] Z.-U.-A. 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.
[51] T. Shahroodi, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "Low-latency double point multiplication architecture using differential addition chain over GF(2^m)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1465–1473, Dec. 2019.

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.

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. In 2004, he joined the Faculty of the School of Information and Electronics, Beijing Institute of Technology. His research interests include image processing and array signal processing, and design and hardware implementation for cryptography.

JINGQI ZHANG received the B.Sc. and M.Sc. degrees in electronic engineering from the Beijing Institute of Technology, Beijing, China, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree. His research interests include digital circuit design, cryptographic computations, and VLSI implementation of cryptosystems.

YIXUAN LUO received the bachelor's degree from the Beijing Institute of Technology, Beijing, China, in 2019, where he is currently pursuing the master's degree. His research interests include digital circuit design, cryptography algorithm, and information security.

SHIWEI REN received the B.Eng. degree in information engineering from the Beijing Institute of Technology, in 2008, and the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences, in 2013. In the same year, she joined the Faculty of the School of Information and Electronics, Beijing Institute of Technology. Her research interests include direction-of-arrival estimation, sparse array signal processing, and digital circuits design.