Received March 18, 2020, accepted April 14, 2020, date of publication April 16, 2020, date of current version May 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988379

Area-Time Efficient Hardware Implementation of Modular Multiplication for Elliptic Curve Cryptography

MD. MAINUL ISLAM[1], (Graduate Student Member, IEEE),
MD. SELIM HOSSAIN[2], (Member, IEEE),
MD. SHAHJALAL[1], (Student Member, IEEE),
MOH. KHALID HASAN[1], (Member, IEEE),
AND YEONG MIN JANG[1], (Member, IEEE)
[1]Department of Electronics Engineering, Kookmin University, Seoul 02707, South Korea
[2]Department of Electrical and Electronic Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh

Corresponding author: Yeong Min Jang (yjang@kookmin.ac.kr)

This research was supported by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) support program (IITP-2018-0-01396) supervised by the Institute for Information and Communication Technology Promotion (IITP).

ABSTRACT In this paper, an area-time efficient hardware implementation of modular multiplication over five National Institute of Standard and Technology (NIST)-recommended prime fields is proposed for lightweight elliptic curve cryptography (ECC). A modified radix-2 interleaved algorithm is proposed to reduce the time complexity of conventional interleaved modular multiplication. The proposed multiplication algorithm is designed in hardware and separately implemented on Xilinx Virtex-7, Virtex-6, Virtex-5, and Virtex-4 field-programmable gate array (FPGA) platforms. On the Virtex-7 FPGA, the proposed design occupies only 1151, 1409, 1491, 2355, and 2496 look up tables (LUTs) and performs single modular multiplication in 0.93 μs, 1.18 μs, 1.45 μs, 2.80 μs, and 4.69 μs with maximum clock frequencies of 207.1 MHz, 190.7 MHz, 177.3 MHz, 137.6 MHz, and 111.2 MHz over five NIST prime fields of size 192, 224, 256, 384, and 521 bits, respectively. The hardware implementations on the Virtex-6, Virtex-5, and Virtex-4 FPGAs also show that the proposed design is highly efficient in terms of hardware resource utilization and area-delay product compared with other designs for modular multiplication.

INDEX TERMS Modular multiplication, interleaved multiplication, elliptic curve cryptography.

I. INTRODUCTION

Elliptic curve cryptography (ECC), a public key cryptography (PKC), has become a buzzword in the fields of network security, digital signatures, and radio frequency identification (RFID) [1]–[3]. It has emerged as an optimum solution for high-speed data encryption and energy efficient node authentication in wireless sensor networks (WSNs) [4]–[6]. The main advantage of ECC is that it uses smaller key size compared with other PKC such as RSA to provide equivalent security strength. ECC can be implemented on both hardware and software platforms, where hardware implementation provides better performance than microcontroller-based software implementation [7]. Hardware-based ECC is more power efficient and can provide significant security improvements compared to software approach. Over the past few years, field-programmable gate array (FPGA)-based ECC schemes have been drawing extensive attention of security researchers by offering a number of advantages over other software-based cryptographic platforms. Their re-programmability, reconfigurability, optimization capability, lower latency, and higher throughput make them more convenient for the IoT security.

The first level of an ECC hierarchy contains finite field arithmetic such as addition, subtraction, multiplication, squaring, and inversion, which can be performed over either a Galois binary field GF(2^m) or a Galois prime field GF(p) [1], where m is the order of the binary field and p is a

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

prime number. Although the hardware implementation of ECC over GF(2^m) is faster than that over GF(p), it is less flexible for variable field sizes [8]. However, multiplication over GF(p), namely, modular multiplication, is a crucial arithmetic operation in ECC. The performance of an elliptic curve cryptosystem is regulated by the efficiency of modular multiplication algorithm. A modular multiplier is an indispensable part of an ECC processor. The area utilization and the maximum achievable frequency of the processor entirely depend on its adopted modular multiplier. It is a major challenge in prime field ECC to develop a low-area modular multiplier for lightweight cryptography without compromising the speed. Hence, optimizing modular multiplication algorithm is a crucial demand. Several hardware implementations of modular multiplication over GF(p) are reported in [8]–[15], where most of the authors focused only on the multiplication speed. They reduced the multiplication time costing more hardware resources, which is not wise when ECC is implemented in IoT platforms. Reducing computation time by expanding area is not the proper way to meet the technical challenges faced by the current cryptographic applications. To fulfill the low latency, low memory, and low energy requirements of resource-constrained IoT devices, designs should be efficient in terms of both area and delay. Lightweight hardware implementations are only solutions for the enhancement of low-area, low-power IoT applications. The details of the designs are explained in a later section of this paper.

This paper presents an efficient interleaved modular multiplication algorithm along with its hardware design and implementations over GF(p) for lightweight, reconfigurable ECC processors [16], where primes p are National Institute of Standard and Technology (NIST)-recommended [17], [18]. The values of p are shown in Table 1. The powers existing in the expressions of these primes are all multiples of 32 except the prime p_{521}. These properties accelerate modular reduction operations on machines with word-size of 32 bits [1].

TABLE 1. NIST-recommended prime fields.

Field	Field size (bits)	Prime value
p_{192}	192	$2^{192} - 2^{64} - 1$
p_{224}	224	$2^{224} - 2^{96} + 1$
p_{256}	256	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
p_{384}	384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
p_{521}	521	$2^{521} - 1$

The remainder of this paper is organized as follows: Section II presents the mathematical background and motivation of this paper. Section III demonstrates proposed algorithm and hardware architectures for modular multiplication. Section IV presents the implementation results of the proposed design. A performance comparison of our proposed modular multiplier with other modular multipliers is shown in Section V. Finally, this research work is concluded in Section VI.

II. MATHEMATICAL BACKGROUND AND MOTIVATION

This section presents the role of a modular multiplier in ECC and comparative discussion about several popular modular multiplication algorithms.

The Weierstrass form of an elliptic curve over GF(p) in affine coordinates is given by the equation:

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

where the variables x, y and the curve constants a, b are in the range $[1, p-1]$ with

$$4a^3 + 27b^2 \neq 0. \tag{2}$$

An ECC processor generates a public key $Q(x, y)$ by multiplying a base point $G(x, y)$ on E_p with a random scalar s such that $Q = s \cdot G$ [19]. This multiplication is called elliptic curve scalar multiplication (ECSM), which is the backbone operation of an ECC processor. It deals with a number of arithmetic and group operations. A modular multiplier multiplies two integers in GF(p) (e.g., $x \times x$, $y \times y$, and $x \times y$) in such a manner that the output always remains in the range $[1, p-1]$. The complexity of ECSM is severely affected by the modular multiplications performed to accomplish the ECSM operation.

Several widely used algorithms for modular multiplication are the Montgomery, interleaved, residue number system (RNS), and the Karatsuba-Ofman [1], [9], [20], [21]. These can be further classified into radix-r ($r = 2, 4, 8\ldots$) computations, which specify the number of bits to be processed in each iteration as well as determine the number of clock cycles required to complete the multiplication operation.

Let q be a prime number in the range of $2^{n-1} < q < 2^n$ such that the greatest common denominator $gcd(2^n, q) = 1$ and A, B be two operands in the range of $[0, q-1]$. The Montgomery algorithm computes the modular multiplication of A and B as $C = A \cdot B \cdot R^{-1}\ mod\ q$, where $R = 2^n$ [22]. This method reduces the delay caused by the trial division required to determine the multiple of the prime q, which would be subtracted from the regular product of the operands. The trial division is replaced by a division by R and a modulo R reduction, which are light operations in finite fields as R is a power of 2 [23], [24]. It is noteworthy that the division by 2^n is exactly the same as n-bit right-shift operation. The main disadvantage of this algorithm is that it needs some preprocessing and post-processing operations to eliminate the extra factor 2^{-n} and confine the intermediate results within the range of the prime field. Furthermore, this algorithm works only on the integers that are represented in the Montgomery domain.

RNS provides high-speed computation by representing a large integer with some small integers and processing the small integers independently. In RNS-based modular multiplication, the prime q is represented by N co-prime moduli such that $\{q_1, q_2, \ldots, q_N\}$ and the operands A, B are represented in N channels such that $A = \{a_1, a_2, \ldots, a_N\}$ and $B = \{b_1, b_2, \ldots, b_N\}$, where $a_i = A\ mod\ q_i$ and $b_i = B\ mod\ q_i$ [25]. Then the multiplication is performed in parallel on the individual residues in the RNS representations of the operands with reduction steps including a division by one of the moduli in each step [14]. The RNS domain is converted back to the original domain by Chinese remainder theorem (CRT) [26]. Although the RNS representations speed up the multiplication process significantly, they consume more power because of their high complexity and require a vast amount of hardware resources to be implemented on machines. Therefore, RNS modular multiplication is not suitable for resource-constrained devices. In the Karatsuba-Ofman method, modular multiplication is performed using divide-and-conquer [1] approach to speed up the multiplication process. Each of the operands is split into smaller segments up to a reasonable size and the segments are multiplied recursively with each other. A major drawback of Karatsuba multiplication is that the amount of resources required to implement this algorithm in hardware grows exponentially with operand size. Although this algorithm is fast, it is highly recursive that increases the hardware complexity [27]. Because of recursive nature, this algorithm is not convenient for low-power, resource-constrained devices.

The basic idea of interleaved modular multiplication is multiply by 2 and add operations followed by modular reduction in a repeated manner. The modular multiplication of the multiplicand X and the multiplier Y over GF(p) is given by the general expression [19],

$$\begin{aligned} Z &= X \cdot Y \; mod \; p \\ &= y_0 \cdot (2^0 X) + y_1 \cdot (2^1 X) + \ldots + y_n \cdot (2^n X) \; mod \; p \\ &= \sum_{i=0}^{n-1} y_i \cdot (2^i X) \; mod \; p, \end{aligned} \tag{3}$$

where p is a prime number, X and Y are two integers such that $X, Y \in [1, p-1]$, and n is the length of the integers.

Algorithm 1 Conventional Interleaved Modular Multiplication (Left-to-Right)

```
Input : $X = \sum_{i=0}^{n-1} x_i 2^i$, $Y = \sum_{i=0}^{n-1} y_i 2^i$;
        $X, Y \in [1, p-1]$ & $x_i, y_i \in \{0, 1\}$.
Output : $Z = (X \cdot Y) \; mod \; p$.
1: $Z \leftarrow 0$;
2: for $i$ from $n-1$ downto 0 do
3:     $U \leftarrow 2Z$;
4:     $V \leftarrow y_i \cdot X$;
5:     $W \leftarrow U + V$;
6:     $Z \leftarrow W \; mod \; p$;
7: end for;
8: return $Z$;
```

Algorithm 1 demonstrates conventional interleaved modular multiplication. In every iteration, the multiplier Y is bitwise multiplied with the multiplicand X in left-to-right direction and added to the partial product Z. Z is reduced to modulo p at the end of each iteration so as to confine the intermediate results always in the range $[1, p-1]$. This algorithm avoids the multiply by 2^n operation required to eliminate the extra factor 2^{-n} exists in the Montgomery algorithm and hence reduces the computation costs of modular multiplication. So far, interleaved modular multiplication is more efficient compared with other multiplications when operand sizes are large.

In order to reduce the time complexity of ECSM over GF(p) as well as the hardware resource requirements for ECC, we propose an area-time efficient modular multiplier, introducing a modified radix-2 interleaved modular multiplication algorithm. The area utilization (in terms of hardware) and the area $\times$ time complexity of the proposed modular multiplier are significantly reduced.

III. PROPOSED ALGORITHM AND HARDWARE ARCHITECTURES

This section presents the proposed interleaved modular multiplication algorithm and hardware architectures for the conventional and the modified interleaved modular multiplications.

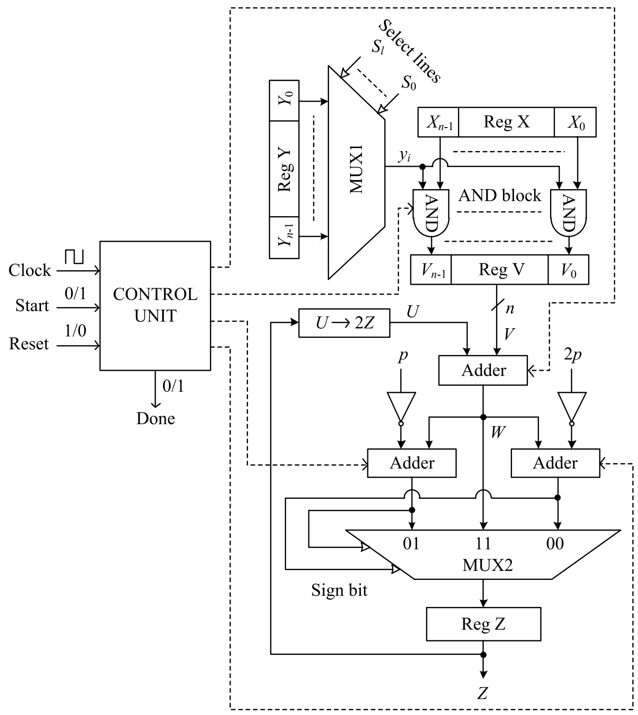

FIGURE 1. Proposed hardware architecture for modular multiplication (Design-1).

A. MODULAR MULTIPLIER (DESIGN-1)

Figure 1 illustrates the proposed hardware architecture of Algorithm 1 that operates iterative bitwise multiplication followed by modular reduction operations. The bitwise multiplication is started from the most significant bit (MSB) of the multiplier Y and moved towards the least significant bit (LSB) of it. The accumulator Z is doubled and computed as U at the beginning of each iteration. The multiplier Y is bitwise multiplied with the multiplicand X and the partial product y_iX is stored in register V (Reg V). The i^{th} bit of Y is selected by multiplexer MUX1 and the multiplication y_iX is performed by n number of 2-input AND gates. U is added to V and the addition is reduced to modulo p in every iteration. The modular reduction is performed by subtracting the prime number p and $2p$ from W. Multiplexer MUX2 is used to store one of the three outputs W, $W-p$, and $W-2p$ into register Z (Reg Z). A total of $n+1$ clock cycles are required to complete this multiplication process.

A major drawback of this design is that it always spends $n+1$ clock cycles to accomplish single modular multiplication even if the multiplier Y is of less than n bits in size.

B. MODULAR MULTIPLIER (DESIGN-2)

Although higher radix (e.g. radix-4, radix-8, radix-16, and radix-32) modular multipliers offer lower latency multiplication than lower radix such as radix-2 modular multiplier, these consume more hardware resources that expand the overall area required by an ECC processor. In order to reduce the area, a modified radix-2 interleaved modular multiplication algorithm is proposed to accomplish right-to-left bitwise modular multiplication. Conventional interleaved modular multiplication is replaced for performing memory-friendly loop operation and reducing the time complexity of multiplication. Algorithm 2 demonstrates the proposed algorithm for modular multiplication and Figure 2 depicts the proposed hardware architecture based on this algorithm.

Algorithm 2 Proposed Interleaved Modular Multiplication (Right-to-Left)

Input : $X = \sum_{i=0}^{n-1} x_i 2^i,\ Y = \sum_{i=0}^{n-1} y_i 2^i;$
$X, Y \in [1, p-1]\ \&\ x_i, y_i \in \{0, 1\}.$

Output : $Z = (X \cdot Y)\ mod\ p.$

```
1:  $Z \leftarrow 0; U \leftarrow 0; V \leftarrow 0; T \leftarrow Y;$
2:  while $Y > 0$ do
3:     if $T = Y$ then
4:        $U \leftarrow X;$
5:     else
6:        $U \leftarrow 2X;$
7:     end if;
8:     if $y_0 = 1$ then
9:        $V \leftarrow Z + U;$
10:    else
11:       $V \leftarrow Z;$
12:    end if;
13:    $Z \leftarrow V\ mod\ p; X \leftarrow U\ mod\ p;$
14:    $Y \leftarrow$ ‘0’ || $Y(n-1$ downto $1);$
15: end while;
16: return $Z;$
```

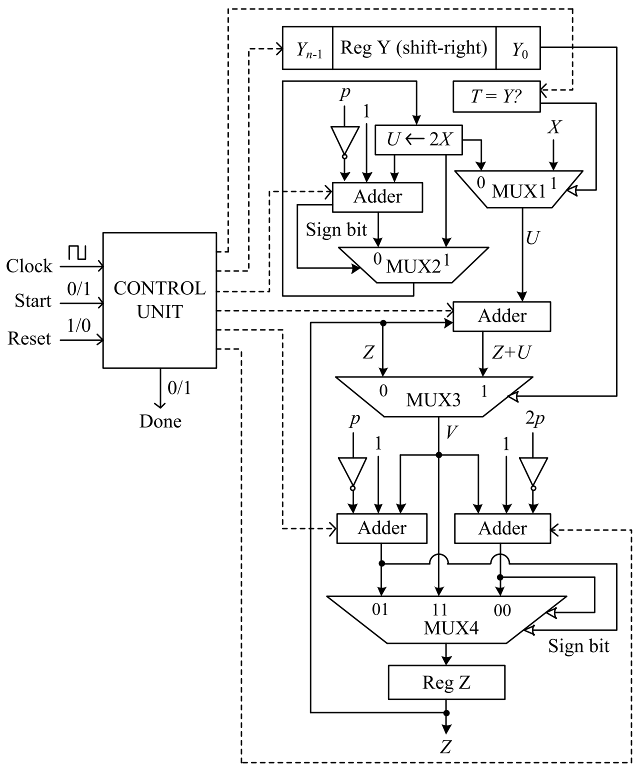

FIGURE 2. Proposed hardware architecture for modular multiplication (Design-2).

In this architecture, the multiplicand X is doubled and added to accumulator Z depending on the least significant bit (LSB) of the multiplier Y. A shift register Y (Reg Y) is shifted right in every iteration for performing right-to-left bitwise multiplication as well as well-synthesized loop operation. In the first iteration, U is computed as X and from the second iteration, it is computed as $2X$. U is added to Z, if $y_0 = 1$ or Z remains unchanged if $y_0 = 0$. Both U and V are reduced to modulo p to confine them in the finite field GF(p) and $V\ mod\ p$ is stored in register Z (Reg Z). For these modular operations, U is subtracted by p and V is simultaneously subtracted by p and $2p$. The subtractions $U-p$, $V-p$, and $V-2p$ are accomplished by 2's compliment method. The comparisons $U \geq p$, $V \geq p$, and $V \geq 2p$ are executed by determining the sign bits of the subtractions $U-p$, $V-p$, and $V-2p$, respectively. At the end of each iteration, register Y is shifted to right by 1 bit. After $n-1$ number of iterations, the multiplier Y is shifted to zero value and the final modular product is obtained from Reg Z.

The design comprises four multiplexers such as MUX1, MUX2, MUX3, and MUX4. MUX1 selects X for the first iteration and $2X$ for the remaining iterations, MUX2 performs $U\ mod\ p$, MUX3 selects Z to keep Z unchanged if $b_0 = 0$ or it selects V to add U to Z if $b_0 = 1$, and MUX4 performs $V\ mod\ p$. The latency to perform single modular

TABLE 2. Implementation results of Design-1 on Virtex-7 FPGA.

Field size (bits)	Clock cycles	Frequency (MHz)	Area (slices)	Area (LUTs)	Time (μs)	Area$\times$Time (LUTs$\times$ms)	Throughput (Mbps)
192	193	147.9	251	972	1.30	1.3	147.1
224	225	143.1	325	1221	1.57	1.9	142.5
256	257	134.7	365	1400	1.91	2.7	134.2
384	385	105.9	578	2048	3.64	7.5	105.6
521	522	89.3	592	2215	5.85	13.0	89.1

TABLE 3. Implementation results of Design-2 on Virtex-7 FPGA.

Field size (bits)	Clock cycles	Frequency (MHz)	Area (slices)	Area (LUTs)	Time (μs)	Area$\times$Time (LUTs$\times$ms)	Throughput (Mbps)
192	193	207.1	386	1151	0.93	1.1	206.0
224	225	190.7	490	1409	1.18	1.7	189.9
256	257	177.3	514	1491	1.45	2.2	176.6
384	385	137.6	820	2355	2.80	6.6	137.2
521	522	111.2	975	2496	4.69	11.7	111.0

multiplication with this design is around $n + 1$ clock cycles, where n clock cycles are spent to accomplish n number of iterations and an extra clock cycle is used for the initialization of the loop operation. The number of clock cycles required for the multiplication depends on the bit length of Y as the shift register used controls over the loop operation. The proposed modular multiplier can act as a modular squarer if the two operands be identical.

The advantage of Design-2 over Design-1 is that it speeds up the multiplication process as the loop operation is executed by a shift register rather than a multiplexer. While performing ECSM, an n-bit ECC processor deals with a number of modular multiplication operations where the bit-length of the operands can be less than n bits. If an ECSM operation includes m successive modular multiplications without any inversion operation, it spends always $m(n + 1)$ clock cycles to be completed according to Algorithm 1. Whereas, the same operation requires less than $m(n + 1)$ clock cycles to be executed when it adopts Algorithm 2. Therefore, Algorithm 2 is more beneficial for ECC than Algorithm 1.

IV. IMPLEMENTATION RESULTS

The proposed modular multipliers were implemented on Xilinx Virtex-7 (XC7VX485T) FPGA platform. For performance comparison with other available modular multipliers, Design-2 was also implemented on Xilinx Virtex-6 (XC6VHX380T), Virtex-5 (XC5VFX130T), and Virtex-4 (XC4VLX160) FPGA platforms, separately. The proposed designs were synthesized, translated, mapped, post placed, and routed on these FPGAs using the Xilinx ISE 14.7 Design Suite. The Xilinx ISim (P.20131013) simulator was used to simulate the modular multiplication and the simulation results were verified by the Maple 18 software. The implementation results of the Design-1 and Design-2 on Virtex-7 FPGA are summarized in Table 2 and Table 3, respectively.

On Virtex-7 FPGA, Design-1 operates at maximum clock frequencies of 147.9 MHz, 143.1 MHz, 134.7 MHz, 105.9 MHz, and 89.3 MHz with 147.1 Mbps, 142.5 Mbps, 134.2 Mbps, 105.6 Mbps, and 89.1 Mbps throughputs for the field sizes of 192, 224, 256, 384, and 521 bits, respectively. The numbers of slices utilized are 251, 325, 365, 578, and 592, which are equivalent to 972, 1221, 1400, 2048, and 2215 look up tables (LUTs). The delays spent to perform modular multiplications over the prime fields $p_{192}, p_{224}, p_{256}, p_{384}$, and p_{521} are 1.3 μs, 1.57 μs, 1.91 μs, 3.64 μs, and 5.85 μs, where the required clock cycles are 193, 225, 257, 385, and 522, respectively.

On the same platform, Design-2 performs 192, 224, 256, 384, and 521-bit modular multiplications in 0.93 μs, 1.18 μs, 1.45 μs, 2.80 μs, and 4.69 μs with maximum clock frequencies of 207.1 MHz, 190.7 MHz, 177.3 MHz, 137.6 MHz, and 111.2 MHz, respectively, where the corresponding throughputs are 206.0 Mbps, 189.9 Mbps, 176.6 Mbps, 137.2 Mbps, and 111.0 Mbps. The design occupies 386, 490, 514, 820, and 375 slices and 1151, 1409, 1491, 2355, and 2496 look up tables (LUTs) to perform these multiplications. Although Design-2 costs a few more hardware resources than Design-1, it offers faster multiplication and is more efficient for ECC. It reduces the latency of the most time-consuming ECSM operation. It should be noted that no additional digital signal processing (DSP) slice is used in these implementations.

V. PERFORMANCE COMPARISON

Several works on FPGA implementation of modular multiplication have been documented in literature, where some authors try to minimize the hardware utilization and others aim to reduce the latency of multiplication. It is a challenging task to achieve a higher processing speed, utilizing fewer hardware resources because area and time are two

TABLE 4. Performance comparison of the proposed modular multiplier with other modular multipliers.

Design	Algorithm	Platform	Field size	Clock cycles	Frequency (MHz)	Area (LUTs)	Time (μ s)	Area$\times$Time (LUTs$\times$ms)
Ours (Design-2)	Radix-2 interleaved (modified)	Virtex-6	192	193	190.4	1183	1.01	1.2
			224	225	173.9	1457	1.29	1.9
			256	257	160.7	1551	1.60	2.5
		Virtex-5	256	257	110.5	2259	2.33	5.3
		Virtex-4	192	193	90.8	1918	2.13	4.1
			224	225	80.9	2323	2.78	6.5
			256	257	75	2563	3.43	8.8
			384	385	55.3	3952	6.96	27.58
			521	522	42.7	5368	12.22	65.6
[8]	Montgomery	Virtex-6	256	50	40.1	23977	0.80	19.2
[9]	Radix-2 interleaved (conventional)	Virtex-II	192	192	40	7328	4.80	35.2
			224	224	37	8547	6.00	51.3
			256	256	34	9821	7.30	71.7
[10]	Regular modular reduction	Virtex-4	192	74	60	31946	1.23	39.3
			224	76	60	31946	1.27	40.6
			256	78	60	31946	1.30	41.5
			384	178	60	31946	2.97	94.9
			521	325	60	31946	5.42	173.1
[11]	Radix-4 Booth encoded	Virtex-6	192	98	101.3	3020	0.97	2.9
			224	114	98.2	3427	1.16	4.0
			256	130	95.2	3877	1.36	5.3
[12]	Karatsuba-Ofman	Virtex-5	256	212	160	19747	1.33	26.3
[13]	Radix-4 interleaved	Virtex-6	256	131	166	6300	0.79	5.0
[14] (Design-1)	Radix-4 Booth encoded interleaved	Virtex-6	192	97	92	11152	1.1	12.3
			256	129	86.6	18520	1.49	27.6
			512	257	76.25	29916	3.37	100.8
[14] (Design-2)	Radix-8 Booth encoded interleaved		192	50	75	16860	0.88	14.8
			256	68	71	22628	0.93	21.0
			512	130	63.86	104480	2.69	281.1
[15] (Design-1)	Radix-4 serial interleaved	Virtex-6	192	48	101	3100	0.94	2.9
			224	56	99	3400	1.13	3.8
			256	64	96	3900	1.30	5.1
[15] (Design-2)	Radix-4 parallel interleaved		192	48	171	4200	0.56	2.4
			224	56	167	4900	0.67	3.3
			256	64	166	5300	0.77	4.1

contradictory parameters of an FPGA-based hardware implementation. We tried to maintain a balance between these two important performance criteria. A performance comparison of our proposed modular multiplier (Design-2) with other modular multipliers is shown in Table 4. Most of the articles have compared area in terms of LUTs instead of slices and some authors have not reported the amount of slices required for their modular multipliers. For this reason area is compared in terms of LUTs in this research study.

Yang *et al.* [8] proposed a low-latency FPGA implementation of 256-bit modular multiplication adopting the Montgomery algorithm. On Virtex-6 FPGA, their multiplication scheme occupies 23,977 LUTs and operates at a maximum frequency of 40.1 MHz. Although the scheme is comparatively expensive in terms of hardware, it performs high-speed multiplication. It takes 0.8 μs to perform 256×256 modular multiplication in 50 clock cycles. Ghosh *et al.* [9] exploited a radix-2 interleaved modular multiplier for ECSM over GF(p). Conventional interleaved modular multiplication was implemented on Xilinx Virtex-II FPGA, costing 7328, 8547, and 9821 LUTs over the prime fields p_{192}, p_{224}, and p_{256}, respectively. Their multiplier requires 4.8 μs, 6 μs, and 7.3 μs to complete modular multiplications over these fields, where the operating frequencies are 40 MHz, 37 MHz, and 34 MHz, respectively. The flexible hardware ECC processor proposed by Ananyi *et al.* [10] supports the five NIST prime fields $p_{192}, p_{224}, p_{256}, p_{384}$, and p_{521}. A regular multiplier is employed for this processor that performs modular reduction at the end of non-modular multiplication instead of performing bitwise multiplication followed by modular reduction.

The multiplier requires a huge memory to store the intermediate output before modular reduction operation. The main feature of this multiplier is that it is flexible, which means that the multiplier can perform 192, 224, 256, 384, and 521-bit multiplications without reconfiguring the hardware. As the design is flexible over different field orders, the hardware utilizations for the multiplications over all the five prime fields are the same. A major drawback of this design is that it performs 192-bit multiplication, costing the same amount of hardware resources as required to perform 521-bit multiplication. The multiplier utilizes 31,946 LUTs with 32 additional DSP slices on Xilinx Virtex-4 FPGA platform and runs at a constant frequency of 60 MHz over all the five prime fields. The clock cycles required to perform modular multiplications over $p_{192}, p_{224}, p_{256}, p_{384}$, and p_{521} are 74, 76, 78, 178, and 325, respectively.

Javeed *et al.* [11] adopted a radix-4 booth encoded modular multiplier for their ECC processor, which is another version of interleaved modular multiplier. In this method, two signed binary numbers are multiplied by 2's complement representations. Instead of adding the multiplier to partial product and quadrupling the partial product, the multiplicand is conditionally multiplied by $\{0, \pm 1, \pm 2\}$ depending on three consecutive bits of the multiplier in right-to-left direction and added to the partial product shifted left by 2 bits. This method gives the same results as the add-and-quadrupling method. However, their multiplier occupies 3020, 3427, 3877 LUTs on Xilinx Virtex-6 FPGA and performs modular multiplications in 0.97 μs, 1.16 μs, and 1.36 μs, operating at frequencies of 101.3 MHz, 98.2 MHz, and 95.2 MHz over the prime fields p_{192}, p_{224}, and p_{256}, respectively. In [12], Marzouqi *et al.* introduced a redundant signed digit (RSD) based iterative Karatsuba multiplier in which three Karatsuba blocks (low, middle, and high) are used to perform bitwise multiplications between two n-bit operands iteratively. Conventional recursive Karatsuba multiplier is modified to iterative-recursive version of it to reduce the hardware complexity. This multiplier consumes 19,747 LUTs on Xilinx Virtex-5 FPGA to perform 256-bit modular multiplication. The operating frequency, clock cycles required, and time to complete the multiplication are 160 MHz, 212, and 1.33 μs, respectively. On the same platform, this multiplier is 1.75 times faster but costs 8.74 times more LUTs than our multiplier. The radix-4 parallel modular multiplier reported in [13] utilizes 6300 LUTs on Xilinx Virtex-6 FPGA and takes 0.79 μs to complete 256-bit modular multiplication, where the maximum clock frequency is 166 MHz. This multiplier performs very low-latency multiplication by parallel multiplication technique. In [14], the authors proposed radix-4 and radix-8 Booth encoded interleaved modular multipliers over general prime fields of 192, 224, and 256-bit field sizes, modifying bit serial interleaved algorithm for modular multiplication. On Virtex-6 FPGA, the radix-4 multipliers occupies 2788, 4630, and 7479 slices, which are approximately equivalent to 11152, 18520, and 29916 LUTs. The times required to perform 192, 224, and 256-bit modular multiplications are 1.1 μs, 1.49 μs, and 3.37 μs. The radix-8 multipliers performs these multiplications in 0.88 μs, 0.93 μs, and 2.69 μs consuming 4215, 5657, and 26120 slices, which are equivalent to 16860, 22628, and 104480 LUTs. Although these multipliers are highly expensive in terms of hardware resources, the number of clock cycles required to perform interleaved modular multiplications significantly reduced due to use the Booth encoding technique. The radix-4 serial and parallel interleaved modular multipliers reported in [15] offer high-speed multiplication by reducing the latency. To perform 192, 224, and 256-bit modular multiplications, the serial multiplier takes 0.94 μs, 1.13 μs, and 1.3 μs with maximum clock frequencies of 48 MHz, 56 MHz, and 64 MHz, whereas the parallel multiplier takes 0.56 μs, 0.67 μs, and 0.77 μs only. The number of LUTs required for these multiplications by the serial multiplier are 3100, 3400, and 3900. On the other hand, the parallel multiplier consumes 4200, 4900, and 5300 LUTs. Figure 3 shows the performance comparison in terms of frequency for variable field sizes. Frequency is a parameter of the hardware implementation of a multiplier that determines the speed at which the module runs. It is directly correlated with the complexity of the hardware design. The frequency of a complex design is lower than that of a simple, memory-friendly design. From this figure, it can be seen that the frequencies of our implementations on different FPGA platforms are higher than that of the other implementations except [12] and [13]. Figure 4 shows the performance comparison in terms of area utilization over different prime fields. From the one-to-one platform-wise comparison, it is observed that our design costs less number of LUTs than any of the other designs. Besides there are no much differences among the hardware requirements of our design for different field sizes. Therefore, the performance

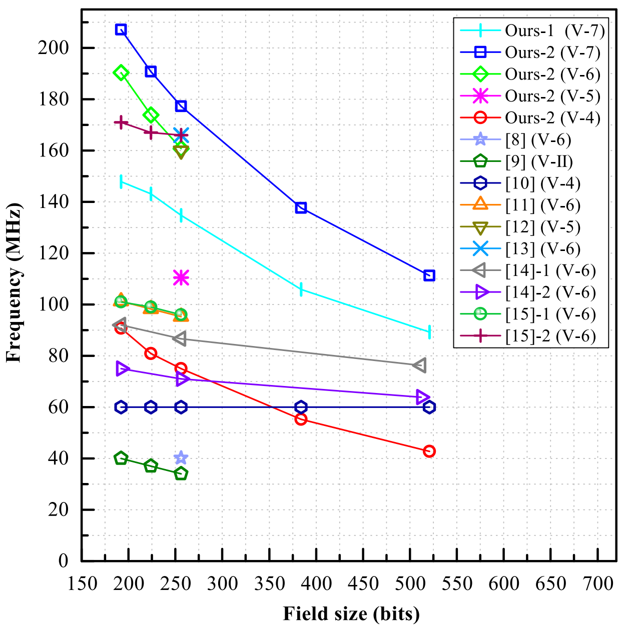

FIGURE 3. Performance comparison in terms of frequency.

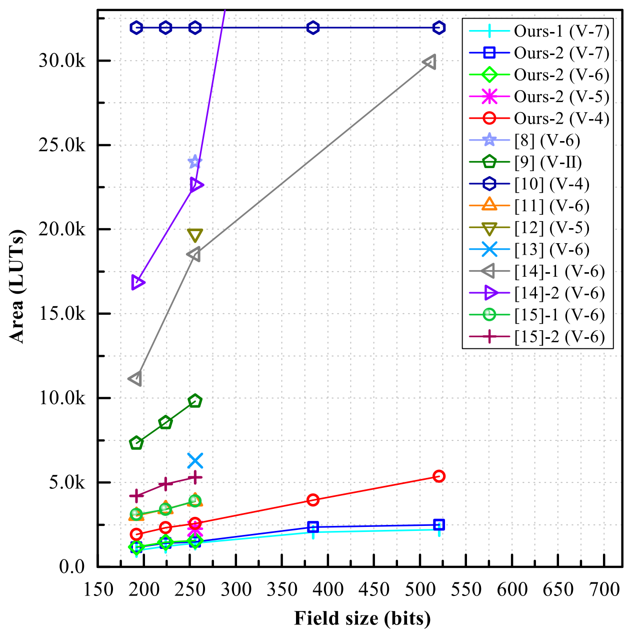

FIGURE 4. Performance comparison in terms of area.

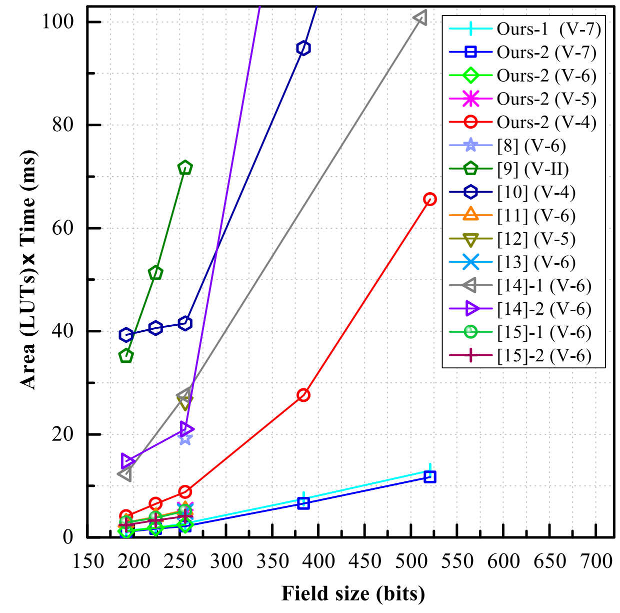

FIGURE 5. Performance comparison in terms of area-time product.

degradation of our proposed modular multiplier for large operands is endurable. Figure 5 shows the performance comparison in terms area-time (AT) product over different prime fields. The performance of a modular multiplier is measured by the reciprocal of its AT product. The smaller the AT product, the better the performance. On the same platform, the AT product of our proposed design is smaller than that of the other designs over the same field, which ensures better performance of our multiplier.

VI. CONCLUSION

A high-performance modular multiplier has been proposed, exploiting a modified interleaved modular multiplication algorithm. The proposed multiplier has been separately implemented on Virtex-7, Virtex-6, Virtex-5, and Virtex-4 FPGA platforms over the five NIST recommended prime fields $p_{192}, p_{224}, p_{256}, p_{384}$, and p_{521}. The implementation results and the performance comparison with other designs show that the area utilization of our design is very low and its area-time product is considerably small. Furthermore, our proposed modular multiplier has diverse application fields in ECC as most of the ECC processors over GF(p) reported in literature are based on five NIST prime curves such as P-192, P-1224, P-256, P-384, and P-521. This multiplier could be recommended to manipulate in low-power, low-memory, resource-constrained cryptographic devices.

REFERENCES

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

[2] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-curve-based security processor for RFID," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008.

[3] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Y. Yang, "Highspeed high-security signatures," *J. Cryptogr. Eng.*, vol. 2, no. 2, pp. 77–89, Sep. 2012.

[4] D. Aranha, R. Dahab, J. López, and L. Oliveira, "Efficient implementation of elliptic curve cryptography in wireless sensors," *Adv. Math. Commun.*, vol. 4, no. 2, pp. 169–187, May 2010.

[5] S. C. Seo and H. Seo, "Highly efficient implementation of NIST-compliant Koblitz curve for 8-bit AVR-based sensor nodes," *IEEE Access*, vol. 6, pp. 67637–67652, 2018.

[6] H. Du, Q. Wen, and S. Zhang, "An efficient certificateless aggregate signature scheme without pairings for healthcare wireless sensor network," *IEEE Access*, vol. 7, pp. 42683–42693, 2019.

[7] 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.

[8] Y. Yang, C. Wu, Z. Li, and J. Yang, "Efficient FPGA implementation of modular multiplication based on Montgomery algorithm," *Microprocessors Microsyst.*, vol. 47, art. 2016, pp. 209–215, Jul. 2016.

[9] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, "Petrel: Power and timing attack resistant elliptic curve scalar multiplier based on programmable GF(p) arithmetic unit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 8, pp. 1798–1812, Jan. 2011.

[10] K. Ananyi, H. Alrimeih, and D. Rakhmatov, "Flexible hardware processor for elliptic curve cryptography over NIST prime fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 8, pp. 1099–1112, Aug. 2009.

[11] K. Javeed, X. Wang, and M. Scott, "High performance hardware support for elliptic curve cryptography over general prime field," *Microprocessors Microsyst.*, vol. 51, art. 2016, pp. 331–342, Jun. 2017.

[12] H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, and T. Stouraitis, "A high-speed FPGA implementation of an RSD-based ECC processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 151–164, Jan. 2016.

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

[14] K. Javeed and X. Wang, "Radix-4 and radix-8 booth encoded interleaved modular multipliers over general F_p," in *Proc. 24th Int. Conf. Field Program. Log. Appl. (FPL)*, Munich, Germany, Sep. 2014, pp. 1–6.

[15] K. Javeed, X. Wang, and M. Scott, "Serial and parallel interleaved modular multipliers on FPGA platform," in *Proc. 25th Int. Conf. Field Program. Log. Appl. (FPL)*, London, U.K., Sep. 2015, pp. 1–4.

[16] C. A. L.-Nino., A. D.-Perez., and M. M.-Sandoval., "Elliptic curve lightweight cryptography: A survey," *IEEE Access*, vol. 6, pp. 72514–72550, Nov. 2018.

[17] E. Barker, "Recommendation for key management—Part 1: General (revision 4)," NIST, Gaithersburg, MD, USA, Tech. Rep. SP 800-57, Jan. 2016.
[18] L. Chen, D. Moody, A. Regenscheid, and K. Randall, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," NIST, Gaithersburg, MD, USA, Tech. Rep. SP 800–186, Oct. 2019.
[19] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, "FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field," *IEEE Access*, vol. 7, pp. 178811–178826, 2019.
[20] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, p. 519, May 1985.
[21] J.-C. Bajard, L.-S. Didier, and P. Kornerup, "An RNS montgomery modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 47, no. 7, pp. 766–776, Jul. 1998.
[22] A. F. Tenca and C. K. Koc, "A scalable architecture for modular multiplication based on montgomery's algorithm," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1215–1221, Sep. 2003.
[23] C. Kaya Koc, T. Acar, and B. S. Kaliski, "Analyzing and comparing montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, Jun. 1996.
[24] Z. Liu and J. Großschädl, "New speed records for Montgomery modular multiplication on 8-bit AVR microcontrollers," in *Progress in Cryptology—AFRICACRYPT* (Lecture Notes in Computer Science), vol. 8469. Berlin, Germany: Springer-Verlag, 2014, pp. 215–234.
[25] S. Asif and Y. Kong, "Highly parallel modular multiplier for elliptic curve cryptography in residue number system," *Circuits, Syst., Signal Process.*, vol. 36, no. 3, pp. 1027–1051, Mar. 2017.
[26] G. X. Yao, J. Fan, R. C. C. Cheung, and I. Verbauwhede, "Novel RNS parameter selection for fast modular multiplication," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 2099–2105, Aug. 2014.
[27] N. Nedjah and L. D. M. Mourelle, "Fast less recursive hardware for large number multiplication using Karatsuba–Ofman's algorithm," in *Computer and Information Sciences* (Lecture Notes in Computer Science), vol. 2869. Berlin, Germany: Springer, 2003, pp. 43–50.

MD. MAINUL ISLAM (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering (EEE) from the Khulna University of Engineering & Technology (KUET), Bangladesh, in 2018. He is currently pursuing the M.Sc. degree in electronics engineering with the Wireless Communication and Artificial Intelligence (WiComAI) Laboratory, Kookmin University, South Korea. His research interests include wireless communications security, cryptography, elliptic curve cryptography (ECC), Edwards-curve digital signature algorithm (EdDSA), the Internet of Things (IoT) security, VLSI design, FPGA technology, blockchain, 5G, and 6G.

MD. SELIM HOSSAIN (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering (EEE) from the Khulna University of Engineering & Technology (KUET), Bangladesh, in 2008, and the Ph.D. degree in engineering from Macquarie University, Sydney, NSW, Australia, in 2017. He was an Intern with Lund University, Sweden, from January 2016 to February 2016. In 2008, he joined the Department of Electrical and Electronic Engineering (EEE), KUET, as a Lecturer, where he became an Assistant Professor, in 2012. In 2019, he joined the Department of EEE, KUET, as an Associate Professor. He has authored or coauthored over 35 refereed journal articles and conference papers. His research interests include cryptosystems (elliptic curve cryptosystem), processor architectures, high-speed and energy-efficient arithmetic circuits, FPGA, ASIC, and antennas. He is also a member of Institution of Engineers, Bangladesh (IEB). In 2013, he received the International Macquarie University Research Excellence Scholarship (iMQRES) to pursue his Ph.D. degree.

MD. SHAHJALAL (Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering (EEE) from the Khulna University of Engineering & Technology (KUET), Bangladesh, in May 2017, and the M.Sc. degree in electronics engineering from Kookmin University, South Korea, in August 2019, where he is currently pursuing the Ph.D. degree in electronics engineering. His research interests include optical wireless communications, wireless security, non-orthogonal multiple access, the Internet of Things, low-power wide-area networks, and 6G mobile communications. He received the Excellent Student Award.

MOH. KHALID HASAN (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Khulna University of Engineering & Technology, Bangladesh, in 2017, and the M.Sc. degree in electronics engineering from Kookmin University, Korea, in 2019. He is currently attached as a full-time Researcher with the WiComAI Lab, Kookmin University. His current research interests include wireless communications, deep learning, 6G, and wireless security. He received the Academic Excellence Award.

YEONG MIN JANG (Member, IEEE) received the B.E. and M.E. degrees in electronics engineering from Kyungpook National University, South Korea, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from the University of Massachusetts, USA, in 1999. He was with the Electronics and Telecommunications Research Institute (ETRI), from 1987 to 2000. Since 2002, he has been with the School of Electrical Engineering, Kookmin University, Seoul, South Korea, where he was the Director of the Ubiquitous IT Convergence Center, from 2005 to 2010, and has also been the Director of the LED Convergence Research Center, since 2010, and the Director of the Internet of Energy Research Center, since 2018. His research interests include 5G/6G mobile communications, Internet of Energy, eHealth, multiscreen convergence, public safety, optical wireless communications, optical camera communication, and the Internet of Things (IoT). He is currently a Life Member of the Korean Institute of Communications and Information Sciences (KICS). He received the Young Science Award from the Korean Government (2003–2006). He has organized several conferences and workshops, such as the International Conference on Ubiquitous and Future Networks (2009–2017), the International Conference on ICT Convergence (2010–2016), the International Conference on Information Networking 2015, and the International Workshop on Optical Wireless LED Communication Networks (2013–2016). He had served as the Founding Chair of the KICS Technical Committee on Communication Networks, in 2007 and 2008, the Executive Director of KICS, from 2006 to 2014, the Vice President of KICS, from 2014 to 2016, and the Executive Vice President of KICS, since 2018. He serves as the Co-Editor-in-Chief of *ICT Express*, which is published by Elsevier. He has been the Steering Chair of the Multi-Screen Service Forum, since 2011, and the Society Safety System Forum, since 2015. He had served as the Chairman of the IEEE 802.15 Optical Camera Communications Study Group, in 2014. He is also serving as the Chairman of the IEEE 802.15.7m Optical Wireless Communications Task Group and the IEEE 802.15 Vehicular Assistive Technology (VAT) Interest Group.