Computers and Electrical Engineering 97 (2022) 107585

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Efficient implementation of Montgomery modular multiplier on FPGA

Ahmed A.H. Abd-Elkader [a,*], Mostafa Rashdan [b,c], El-Sayed A.M. Hasaneen [d], Hesham F.A. Hamed [e,f]

[a] Qussier Telecom, Telecom Egypt, Red Sea, Egypt
[b] College of Engineering and Technology, American University of the Middle East, Kuwait
[c] Faculty of Energy Engineering Aswan University, Aswan, Egypt
[d] Faculty of Engineering Aswan University, Aswan, Egypt
[e] Faculty of Engineering Minia University, Minia, Egypt
[f] Faculty of Engineering, Egyptian-Russian University, Cairo, Egypt

ARTICLE INFO

Keywords:
Efficient implementation
Nexys-3
FPGA
Montgomery modular multiplication
AT/b
Virtex-6

ABSTRACT

Recent developments in embedded devices have enhanced the demand for systems using compact cryptographic modules. Modular multiplication with large modulus is a central procedure in many public-key cryptosystems. This work describes a creative structure of an FPGA hardware architecture to compute the modular multiplication of two integers. The proposed design has enhanced the hardware structure of Montgomery Modular Multiplier (MMM) in a more efficient way to increase the performance and decrease the area cost. The proposed design is coded in VHDL, and implemented in real-time on the Nexys-3 board. The synthesis results of implementing the proposed 256-bit and 1024-bit modular multiplier on Virtex-6 FPGA have a computation time 1.79 μs and 20.53 μs, occupies 1104 LUTs and 4450 LUTs, and runs at 143.82 MHz and 49 MHz, respectively. Compared to other related FPGA implementations, the proposed design uses less FPGA resources with better efficiency and lower Area Time per bit-length (AT/b).

1. Introduction

Cryptography has a pivotal role in information security [1]. Recently, a considerable literature has grown up around the theme of securing the Internet of Things (IoT) technologies and its applications as a central constituent for electronic devices in the form of the embedded and System-on-Chip (SoC) [2]. The transformation of sensitive data between the devices of IoT in various industrial, medical, commercial and military sectors is a critical issue, and presently it is one of the main characteristics delaying the deployment of pervasive computing environments [3]. For the security schemes that based on public key cryptography, there is a crucial need to utilize hardware modules to achieve a high throughput. Thence, existing researches recognize the vital role played by the development of hardware cryptographic modules. Security delivered by public key encryption algorithms required many arithmetical operations on

This paper was submitted for regular issues, but it is for special section VSI-fpga3. Reviews were processed by Associate Editor Dr. Jiafu Wan and recommended for publication.

* Corresponding author.
E-mail addresses: eng.ahmed_a_elkader@yahoo.com (A.A.H. Abd-Elkader), Mostafa.Rashdan@aum.edu.kw (M. Rashdan), hasaneen@aswu.edu.eg (E.-S.A.M. Hasaneen), hfah66@yahoo.com (H.F.A. Hamed).

https://doi.org/10.1016/j.compeleceng.2021.107585
Received 1 April 2020; Received in revised form 9 July 2021; Accepted 26 October 2021
Available online 20 November 2021

abstract algebraic structures (groups and finite fields) [4]. In addition, these operations are performed conventionally over large numbers (160–3072 bits), making them very time-consuming. That issue has inspired the researchers to invent new hardware dedicated to accelerate the computation time as the focal design goal, which comes at the cost of high hardware resource consumption. Nevertheless, the main problem with such designs is that the embedded systems require fewer physical resources. As a result, this is one of the biggest challenges in implementing light-weight cryptographic systems [2].

The most applied asymmetric cryptographic algorithms such as the Digital Signature Algorithm (DSA), RSA, and Elliptical Curve Cryptography (ECC) depend mainly on modular multiplication [4]. Thus, a high-performance cryptographic system relies upon the construction of modular multiplication. The commonly used modular multiplier is a Montgomery Modular Multiplier (MMM) [5]. MMM is an effective technique for the high-performance hardware implementation of the modular multiplication with large operands [6–10]. This algorithm replaces the trial division with a sequence of additions and right shift operations.

This paper provides a new technique of implementing high efficiency modular multiplier with a few resources, which is a consequential aspect in the implementation of systems on the embedded devices. Implementation of cryptographic algorithms on Field Programmable Gate Array (FPGA) is the preferred solution. The FPGA is low-cost, flexible and long-term maintenance. Especially for cryptographic systems; FPGA is flexible to be reconfigured for any new security demands. Low-cost and low-power FPGAs are obtainable in the market, and it is expected they become popular for applications such as wireless sensor network (WSN) or the Internet of Things (IoT) [2,11–13]. The provided algorithm is an enhancement of MMM algorithm to reduce the area and improve the efficiency of MMM. The new algorithm has been coded in VHDL, and targeted Xilinx Virtex-6 xc6vsx475t-2ff1156 FPGA platform. Xilinx ISE 14.4 has been used to synthesize the proposed design. The ModelSim has been used as a simulation tool. The proposed design has been implemented in real-time using Spartan-6 FPGA on Nexys-3 board. The proposed multiplier different bit-length (160, 192, 224, 256, 384, 512, 1024) have been compared with the related designs in terms of maximum frequency in MHz, area in Slice LUTs, time in μs, throughput in Mbps, Area-Time per bit-length (AT/b) and efficiency (Mbps per LUT). A comparison of the results between the proposed design and other related works reveals the improvement of the consumed hardware resources of the proposed design over other designs. The proposed multiplier computes a 256-bit and 1024-bit modular multiplication in 1.79 μs and 20.53 μs, occupies 1.1 K LUTs and 4.5 K LUTs, and runs at 143.8 MHz and 49.9 MHz, respectively.

The rest of the paper is organized as follows. Section 2 explorers the classical algorithm of Modular Multiplier Reduction, the original MMM, and radix-2 MMM. Section 3 provides the algorithm and hardware architecture of the proposed design. Section 4 describes the performance analysis of the proposed design. The simulation of the proposed design, and the accuracy of the design has been verified using ModelSim. The section provides the synthesized results of implementation of the proposed design on XILINX Virtex-6 FPGA utilizing Xilinx ISE, and describes the comparison with related works. Finally, Section VI concludes our contribution work.

2. Preliminaries

2.1. Basic algorithm of modular multiplier

Algorithm 1 represent a direct approach to calculate the modular multiplication of two integers A and B. At first the product α of the integer numbers is obtained. The modular reduction arithmetic generally involves a division process D of α by the modulus m to obtain the quotient q and residue P. Modular arithmetic and the computations of residue p when α is divided by m is presented in [14].The last step is obtaining the residue P of the division operation as a result of the modular multiplication reduction. It is a very time-consuming operation on both software and hardware platforms.

2.2. Algorithm of Montgomery modular multiplier

Montgomery presented a technique to avert the division process of the two integers modular multiplication [5]. For integers A, B, and$m$, Montgomery replaced the modulus m with the modulus R in the modular reduction process. The Montgomery technique to compute the multiplication reduction with modulo m for two integers A and B is as follows.

Choosing $R > m$. R is an integral power of 2. The integer m is odd to satisfy the condition of the Greatest Common Divisor $gcd(R, m) = 1$.

Precompute the integers R^{-1}*and* $\widetilde{\mathrm{m}}$ such that:

$$RR^{-1} - m\widetilde{m} = 1 \tag{1}$$

$$R^{-1} \equiv 1 \ mod \ m \tag{2}$$

$$\widetilde{m} = -m^{-1} \ mod \ R \tag{3}$$

Transform the operand to its Montgomery domain, which is known also with *REDC* function [5].

$$REDC(x, y) = x \times y \times R^{-1} mod \ m \tag{4}$$

$$\overline{\mathrm{A}} = REDC\left(\mathrm{A}, R^2\right) = \mathrm{A} \times R \ mod \ m \tag{5}$$

$$\overline{\mathrm{B}} = REDC\left(B, R^2\right) = B \times R \bmod m \tag{6}$$

Giving the product in Montgomery domain, such that:

$$\overline{P} = REDC\left(\overline{A}, \overline{B}\right) = \mathrm{A} \times \mathrm{B} \times \mathrm{R} \bmod m \tag{7}$$

$$\text{If } \overline{P} > \mathrm{m} \text{ then } \overline{P} = \overline{P} - m \tag{8}$$

The preceding step is costly because of its modular reduction using modulus m. Therefore, Montgomery used a more practical method to calculate it.

$$REDC(x, y) = \frac{(\mathrm{S} + \mathrm{S}\ m \bmod R\ )m}{R} \tag{9}$$

Where $S = x \times y$. This equation is applied for $\overline{A}, \overline{B}, \overline{P}$.

To obtain the final result, the Montgomery domain transformation must be inversed to its original form:

$$REDC\left(\overline{P}, 1\right) = \frac{\left(P + \overline{P}\ \widetilde{\mathrm{m}} \bmod R\ \right)m}{R} \tag{10}$$

The preceding method is not fast because of a lot of multiplication and addition operations. More effective technique is presented in Algorithm 2. Algorithm 2 represent *radix* $-$ r Montgomery Modular Multiplier algorithm (denoted by MMM algorithm). MMM algorithm utilizes the *REDC*function for two integer operands $\mathscr{A}$, $\mathscr{B}$ directly. Where $\mathscr{S} = \mathscr{A} \times \mathscr{B}$.

$$P = REDC(\mathrm{A}, \mathrm{B}) = \frac{(\mathrm{S} + \mathrm{S}\ \widetilde{\mathrm{m}}\ mod\ R\ )m}{R} \tag{11}$$

MMM algorithm has k iterations. k is the bit length of the modulus m. The multiplier A are scanned from LSB to MSB. $\mathrm{P}_0 and \mathrm{B}_0$ are the LSBs of P*and*B, respectively. The Steps 3 and 4 is repetitive for every iteration for its corresponding A_i and its accumulated value P. According to the Step 3, t_i result is 1 or 0, the modulus m are added or not in Step 4. To get the validated result, the MMM algorithm is repeated again.

$$REDC(P, 1) = \frac{(P + P\ \widetilde{\mathrm{m}}\ mod\ R\ )m}{R} \tag{12}$$

Algorithm 2 represent the radix-rversionofMMM, and Algorithm 3 represent the radix-2 [15,16]. Where $r = 2$, and $\widetilde{m} = 1$.

Division by 2 in Step-4 is performed by shifting the result one-bit to right. Two additions are used for each loop, beside the final subtraction.

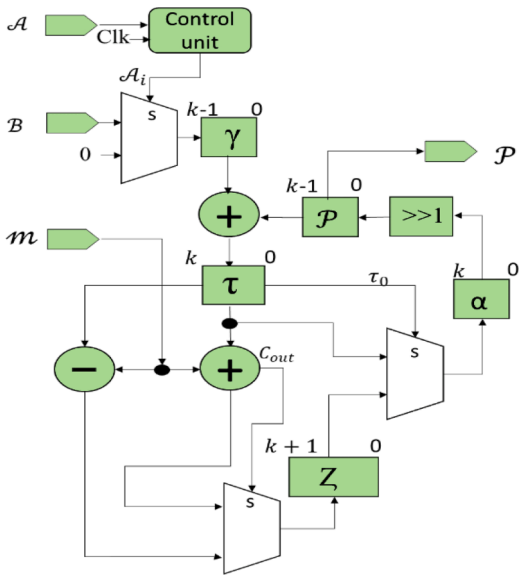

Fig. 1. Proposed MMM Architecture.

3. MMM proposed design

3.1. MMM proposed algorithm

The enhanced algorithm of MMM radix-2 modular multiplication is introduced in Algorithm 4. This algorithm is the enhancement of Algorithm 3.A_i in Algorithm 3 is the multiplier corresponding bit of iteration *i*. A_i is multiplied by the multiplicand LSB B_0 in Step-3, and by the multiplicand B in Step-4. The multiplication in Step-3 of Algorithm 3 is implemented by AND gates, and in Step-4 implemented by a Multiplexer. In the Step-3 of Algorithm 4, $A_i$is used only once to select between two values zeros or the multiplicand B. The Multiplexers, Adder, and Subtractor of modulus m in Steps-3 and 6 of Algorithm 3 have been replaced and rearranged in Steps-5 and 6 of Algorithm 4. The additions and right-shift operations in Step-4 of Algorithm 3 have been separated into multi operations as a pipelined structure in Steps 4 to 7 of Algorithm 4 to improve the performance.

As a result, these modifications and techniques enhance the proposed circuit performance. Additionally, the cost of the hardware resources and interconnects required for the circuit will be less. Thus, the area consumed is small, which is the key aspect of the lightweight cryptographic systems.

3.2. Hardware architecture of the proposed design

The hardware architecture of Algorithm 4 is shown in Fig. 1. The proposed circuit applies a counter instead of using loop to reduce the hardware cost, it requires one clock for one iteration. The clock adds one to the accumulated value of the counter, and the counter counts up to *k* iterations. Due to the registers that are used in the pipelining operations, there is a delay of two clocks. The counter is controlled by a control unit which scans the multiplier A from LSB to MSB to provide the corresponding bit of the iteration ($\mathscr{A}_i$). The proposed circuit is designed of one *k*-bit Adder, one $k+1$ bit Adder, one $k+1$bit Subtractor, and three Multiplexers.

The first step is setting the register *P* to zero. The corresponding bit $\mathscr{A}_i$ choose one of two values, the multiplicand $\mathscr{B}$ if it is '1', or zero $k-bits$ if it is '0'. The output of the first Multiplexer is denoted by γ. The first Adder executes the addition of the accumulated value *P* and the corresponding value of register γ. The output is set to register τ. The second Adder executes the addition of τ and modulus *m*. In Step (5), the register Z must keep its size in $K+1$ bits for the second step. If the carry out (C_{out}) of Adder ($\tau+m$) is '0' then the register Z will be set to the value of the Adder, otherwise will be set to the output of the subtraction process τ- *m*, both operations of addition and subtraction are executed in parallel. The second Multiplexer chooses between the outputs of the second adder or the subtractor to set the value of register Z.

The LSB of the register τ defines if it is odd or even. If LSB of τ is '1' then τ is odd, otherwise it is even. In the Step 7, the output of the third multiplexer must be even. In case τ is even, the value of α will be set to τ value, otherwise $\alpha = Z$. According to the condition that modulus *m* must be odd, the result of odd integer $\pm m$ will be always an even integer. In Step 5, if τ is odd it converted to even by adding modulus *m* to it or by subtracting modulus *m* from it and loaded to register Z. The third Multiplexer selects an even value of Z or τ and the output is set to variable α. The value of α is shifted one bit to right, divided by 2, and loaded to the register of the accumulated value *P*. Those steps are repeated for each clock. Thus, the proposed hardware architecture takes $k+2$ clock cycle to carry out the calculation process. The output is the reduction product of the two input integers multiplied by 2^{-k}. The bit-size of the circuit components relies on the number of operands and modulus bits of the modular multiplication process. This design can be easily configured for any required size of bits.

4. Performance analysis

The MMM circuit architectures for existing and proposed method have been presented in Sections 2 and 3, respectively. The circuit of the proposed design has been coded using VHDL. This work has been synthesized and simulated using Xilinx ISE and ModelSim

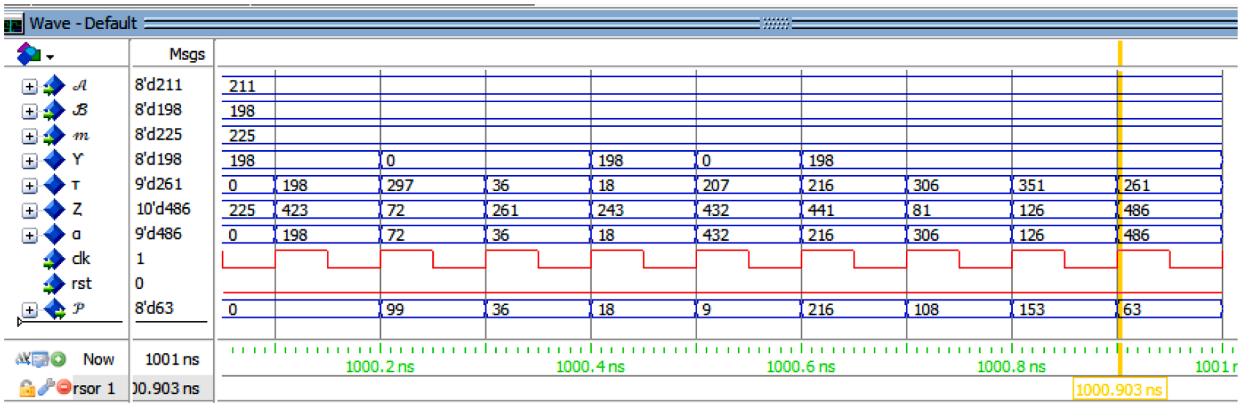

Fig. 2. Simulation of the proposed MMM.

tools, respectively. Where the target FPGA is Xilinx Virtex-6 XC6VSX475T-2FF1156.

4.1. Verification of MMM proposed design

The simulation of the proposed design has been carried out using ModelSim. The verification and simulation results are shown in Fig. 2. As an example, $\mathscr{A} = 211 = 11010011_2, \mathscr{B} = 198$, and the modulus $m = 225$. As expected, the final result is $\mathscr{P} = 63$,where $\mathscr{P} = \mathscr{A} \times \mathscr{B} \times 2^{-k} \; mod \; m = 211 \times 198 \times 2^{-8} \; mod \; 225 = 211 \times 198 \times 196 \bmod 225 = 63$. This example is shown as a numerical example of the proposed algorithm in the below steps.

$P = 0$

1- When $i = 1$

A_1='1'
$\gamma = \mathscr{B} = 198$
$\tau = P + \gamma$=198,
$C_{out(\tau + m)} = '0'$, Z=$\tau + m = 423$
τ_0= '0', $\alpha = \tau$=198,
$P = \alpha /2 = 99$
2- When $i = 2$
A_1='1'
$\gamma = \mathscr{B} = 198$
τ = 99+198=297
$C_{out(\tau + m)} = '1'$, Z=$\tau - m = 72$
τ_0= '1', α = Z =72,
$P = \alpha /2 = 36$
3- When $i = 3$
A_1='0'
$\gamma = 0$
τ = 36+0 = 36
$C_{out(\tau + m)} = '0'$, Z=$\tau + m = 261$
τ_0= '0', $\alpha = \tau = 36$
$P = \alpha /2 = 18$
4- When $i = 4$
A_1='0'
$\gamma = 0$
τ = 18+0 = 18
$C_{out(\tau + m)} = '0'$, Z=$\tau + m = 243$
τ_0= '0', $\alpha = \tau = 18$
$P = \alpha /2 = 9$
5- When $i = 5$
A_1='1'
$\gamma = 198$
$\tau = 9 + 198$=207
$C_{out(\tau + m)} = '0'$, Z=$\tau + m = 432$
τ_0= '1', α = Z = 432
$P = \alpha /2 = 216$
6- When $i = 6$
A_1='0'
$\gamma = 0$
$\tau = 0 + 216$=216
$C_{out(\tau + m)} = '0'$, Z=$\tau + m = 441$
τ_0= '0', $\alpha = \tau = 216$
$P = \alpha /2 = 108$
7- When $i = 7$
A_1='1'
$\gamma = 198$
τ = 198+108=306
$C_{out(\tau + m)} = '1'$, Z=$\tau - m = 81$
τ_0= '0', $\alpha = \tau = 306$

$P = \alpha /2 = 153$
8- When $i = 8$
A_1='1′
$\gamma = 198$
$\tau = 198+153=351$
$C_{out(\tau + m)} = '1'$, Z=$\tau - m = 126$
τ_0= '1′, α = Z = 126
$P = \alpha /2 = 63$

The result is as expected from the calculation and the previous simulation.

The verification of the proposed design using the preceding example was carried out using Xilinx Spartan-6 XC6SLX16-CSG324C on Nexys 3 board, Fig. 3. The slide switches have been used as an input port of the operands, *H, y*, and modulus *n*. The right, the left, and the center push buttons used to enable the inputs *H, y* and *n*, respectively. The 7-segement display has been used to display the inputs and the outputs in hexadecimal. As an example, $k = 8, H = \mathscr{A} = 211_{10} = D3_{16}, y = \mathscr{B} = 198_{10} = C6_{16}$, and the modulus $m = 225_{10} = E1_{16}$. The result is $P = 3F_{16} = 63_{10}$. The result of the implementation is matching the calculation and the simulation.

4.2. *Implementation of the proposed circuit and comparison with related designs*

This section presents a comparison of state of the art *GF*(*p*)hardware Montgomery Modular Multiplication. Table 1 shows the synthesis results of the proposed hardware architecture and the existing related designs. The metrics which are used to evaluate the designs in the comparison for various bit-lengths are area in Slice LUTs, maximum frequency in MHz, time in μs, throughput in Mbps, Area-Time per bit-length (AT/b), and efficiency (Mbps per FPGA LUT). Efficiency metric has been suggested in previous works to evaluate both of the area resources and performance achieved in cryptographic hardware architectures [17].

Javeed et al. [16] provided an FPGA implementation of radix-2 IMM and MMM algorithms. The authors show that the design of radix-2 MMM is more effective in terms of FPGA slice area, calculation time, and throughput as compared to the design of radix-2 IMM. The synthesis results of implementing MMM are shown in Table 1. Javeed et al. [[16] provides a lower computation time and higher throughput, but at the cost of high area. Therefore, this work possessed a higher AT/b requirement and lower efficiency. Our work has accomplished an improvement in efficiency by 30%, while consuming only about 73% of the total slice-LUTs when compared with Javeed et al. [[16].

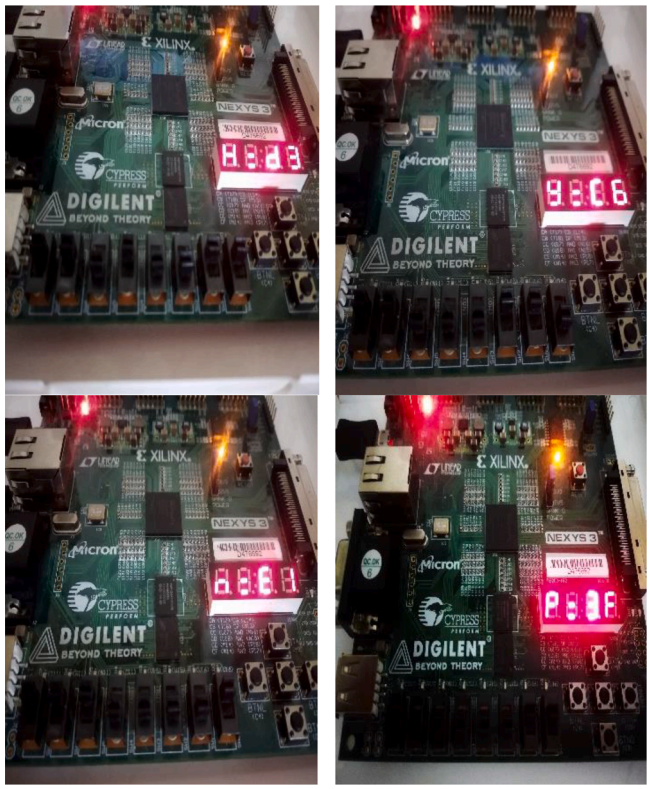

Fig. 3. Verification of the of the proposed MMM on Nexys 3 board.

Table 1
Comparison of modular multiplication implementations on Virtex-6 FPGA.

Design	Bit-length	Frequency (MHz)	Slice LUTs	DSP	Normalized LUT	Time (μs)	Throughput (Mbps)	AT/b	Efficiency (Mbps/LUT)
[16]	160	207	958	0	958	0.78	205.13	4.67	0.214
	192	186	1150	0	1150	1.04	184.62	6.23	0.161
	224	169	1342	0	1342	1.34	167.16	8.03	0.125
	256	154	1534	0	1534	1.68	152.38	10.07	0.099
	384	115	2302	0	2302	3.36	114.29	20.14	0.050
	512	92.5	3070	0	3070	5.56	92.09	33.34	0.030
	1024	51	6142	0	6142	20.10	50.95	120.56	0.008
[18]	160	191	2002	0	2002	0.84	190.48	10.51	0.095
	192	184.6	2401	0	2401	1.04	184.62	13.01	0.077
	224	179.1	2787	0	2787	1.25	179.20	15.55	0.064
	256	174	3207	0	3207	1.48	172.97	18.54	0.054
[19]	160	91.6	2911	0	2911	0.88	181.82	16.01	0.062
	192	89	3511	0	3511	1.08	177.78	19.75	0.051
	224	87.3	4053	0	4053	1.29	173.64	23.34	0.043
	256	85.5	4606	0	4606	1.50	170.67	26.99	0.037
[23]	256	206	16,900	108	81,376	0.13	1954.20	41.65	0.024
[24]	256	166	5300	0	5300	0.77	332.47	15.94	0.063
[25]	256	68	187,900	0	187,900	0.01	17,414.97	10.79	0.093
The proposed Design	160	187	696	0	696	0.87	3.81	182.99	0.262
	192	170	832	0	832	1.15	5.01	166.58	0.200
	224	156	968	0	968	1.47	6.37	152.83	0.157
	256	143	1104	0	1104	1.79	7.74	142.71	0.129
	384	109	1677	0	1677	3.56	15.59	107.76	0.064
	512	88	2231	0	2231	5.87	25.64	87.15	0.039
	1024	49	4450	0	4450	20.68	89.98	49.52	0.011

Algorithm 1.
Basic modular multiplier.

INPUT: Integers (A, $\mathscr{B}$, m)
OUTPUT: $P = A \times Bmodm.$
1. $\alpha = \mathscr{A} \times \mathscr{B}$;
2. $\mathscr{D} = \frac{\alpha}{m} = qm + \mathscr{P}$;
3. Return ($\mathscr{P}$).

Algorithm 2.
Montgomery Modular Multiplier (MMM).

Input: Integers $(\mathscr{A}, \mathscr{B}, m)[k$ bits] radix$r$ representation.
Where $0 \leq (\mathscr{A}, \mathscr{B}) < m$, $R = r^k, gcd(m, r) = 1$,
$\widetilde{m} = -m^{-1} \bmod R$
Output: $\mathscr{P} = \mathscr{A} \times \mathscr{B} \times R^{-1} \ mod \ m$
1. $\mathscr{P} = 0$;
2. For(ifrom0tok $-$ 1, $i = i + 1$)**do**
3. $t_i = ((\mathscr{P}_0 + \mathscr{A}_i \times \mathscr{B}_0) \times \widetilde{m}) \bmod r$;
4. $\mathscr{P} = (\mathscr{P} + \mathscr{A}_i \times \mathscr{B} + t_i \times m)/r$;
5. **Loop**;
6. **If** $\mathscr{P} > m$ **then** $\mathscr{P} = \mathscr{P} - m$; **end If**
7. **Return**(P).

Ghosh et al. [18] presented an alternative design of an interleaved multiplier with Montgomery ladder and high-speed adder circuits. The proposed design of Ghosh et al. [18] provides higher throughput and frequency. By contrast, this work consumes much higher hardware resources, has a lower efficiency and a higher AT/b requirement. A comparison of the two works indicates that our proposed circuit has attained an enhancement in efficiency by 2.33 times the efficiency of Ghosh et al. [18], while consuming only around one-third of the total slice-LUTs.

Javeed and Wang [19] introduced a modified design of Booth encoded radix-4 and radix-8 interleaved modular multipliers over general $GF(p)$. The design introduced by Javeed and Wang [19] has a lower clock- cycles, and higher throughput due to the utilization of radix-4 and radix-8 approach. In contrast, this design utilized much hardware resources, lower efficiency, and a higher AT/b

Algorithm 3.
Radix-2 Montgomery Modular Multiplier.

Input: Integers $(\mathscr{A}, \mathscr{B}, m)[k\ bits]$ radix-2 representation.
where
$0 \leq (\mathscr{A}, \mathscr{B}) < m,\ R = 2^k, \gcd(m, 2) = 1.$
Output: $\mathscr{P} = \mathscr{A} \times \mathscr{B} \times 2^{-k}\ mod m$
1. $\mathscr{P} = 0$;
2. For(ifrom0tok $-$ 1, $i = i + 1$)**do**
3. $t_i = (\mathscr{P}_0 + \mathscr{A}_i \times \mathscr{B}_0)\ mod\ m$;
4. $\mathscr{P} = (\mathscr{P} + \mathscr{A}_i \times \mathscr{B} + t_i \times m)/2$;
5. **Loop**;
6. **If** $\mathscr{P} > m$ **then** $\mathscr{P} = \mathscr{P} - m$; **end If**
7. **Return**(P).

Algorithm 4.
Proposed MMM algorithm.

Input: Integers (A, B, m)[$kbits$] radix 2 representation.
where $0 \leq (A, B) < m, \gcd(m, 2) = 1.$
Output: $\mathscr{P} = \mathscr{A} \times \mathscr{B} \times 2^{-k} \mod m$
1. $P = 0$;
2. *For*(*ifrom*0*tok* $-$ 1, $i = i + 1$)*do*
3. If $\mathscr{A}_i$= '1' then
$\gamma = \mathscr{B}$;
Else
$\gamma = 0$;
End If;
4. $\tau = P + \gamma$;
5. If $C_{out}(\tau + m)$ ='0' then
Z= τ+ m;
Else
Z= τ - m;
End If;
6. If τ_0= '1' then
α = Z;
Else
$\alpha = \tau$;
End If;
7. $P = \alpha$ >>1;
8. Loop;
9. Return P

requirement. Our proposed algorithm has achieved an improvement in efficiency by 3.4 times the efficiency of Javeed and Wang [19]. Furthermore, our design consumed almost 24% of the total slice-LUTs [19].

The related studies offered the term "Normalized-LUT" to stand for DSP cost in the measure of LUT [20–22]. Yan et al. [23] offered an FPGA-based design of hybrid 256-bit MMM over $GF(p)$. This design used the multiplication algorithms of Knuth and Karatsuba in different stages. The work of Yan et al. [23] provided high frequency and throughput over our proposed work. By contrast, the area cost was very high, and the efficiency was very low. Our work has accomplished an enhancement in efficiency by 5.24 times the efficiency of Yan et al. [23], while consuming only 1.38% of the total slice-LUTs.

Based on an interleaved multiplier algorithm and Montgomery power laddering, Javeed et al. [24] offered a design of radix-4 modular multiplier. This work has an improvement in frequency and throughput over the previous work in [19] by twice. Nevertheless, our proposed hardware circuit has a progressive efficiency over the design in [24] by twice, even though consuming only 21.23% of the total slice-LUTs.

Liu and Li [25] introduced a design of 258-bit multiplier based on KO-3 and Karatsuba algorithms. This design was better in terms of time and throughput than our proposed work. Nevertheless, the design of Liu and Li [25] has much higher area than our proposed work. Therefore the design of Liu and Li [25] having a lower efficiency and higher AT/b requirement. Our work has attained an advance in efficiency by 1.6 times the efficiency of Liu and Li [25], while consuming only 0.6% of the total slice-LUTs.

5. Conclusion

This paper has presented an efficient design of Modular Multiplication algorithm to perform the modulo operation in finite fields. The proposed architectures are based on Montgomery Multiplication algorithms.

The new design employed the internal signal to be processed using the pipelining technique. The final subtraction is processed is

implemented to be calculated in parallel with the internal addition to decrease the delay and enhance the performance. This work has been implemented in VHDL, synthesized on Xilinx Virtex-6 FPGA platform, and verified by Modelsim simulator. The related works in the literature are evaluated on the basis of FPGA resource consumption, frequency, computation time, throughput, AT/b, and efficiency. The new design saves around 26% of hardware resources of the most compact related work. From the synthesis results, we can conclude that the proposed multiplier consumes the smallest area as compared to other related multiplier designs. As a result, the developed architecture has efficient area-time compared with other related works in the literature. Therefore, the developed algorithm is a very suitable candidate for making asymmetric cryptosystems efficient compact modules. In the future, we will try to further improve the speed of the multiplier by reducing critical path delay.

CRediT authorship contribution statement

Ahmed A.H. Abd-Elkader: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Mostafa Rashdan:** Conceptualization, Validation, Formal analysis, Investigation, Data curation, Writing – review & editing, Visualization. **El-Sayed A.M. Hasaneen:** Conceptualization, Validation, Formal analysis, Investigation, Data curation, Writing – review & editing. **Hesham F.A. Hamed:** Conceptualization, Validation, Formal analysis, Investigation, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Delfs H, Knebl H. Introduction. 3rd ed. Berlin Heidelberg: Springer Berlin Heidelberg; 2015. p. 1–10. https://doi.org/10.1007/978-3-662-47974-2_1.
[2] Sklavos N, Zaharakis ID. Cryptography and security in Internet of Things (IoTs): models, schemes, and implementations. In: Proceedings of the 8th IFIP international conference on new technologies, mobility and security (NTMS). Larnaca: IEEE; 2016. p. 1–2.
[3] Miorandi D, Sicari S, Pellegrini FD, Chlamtac I. Internet of things: vision, applications and research challenges. Ad Hoc Netw 2012;10:1497–516. https://doi.org/10.1016/j.adhoc.2012.02.016.
[4] Stallings W. Cryptography and network security: principles and practice. 6th ed. Pearson Education; 2014. six.
[5] Montgomery PL. Modular multiplication without trial division. Math Comput 1985;44:519. https://doi.org/10.2307/2007970.
[6] Ozcan E, Erdem SS. A high performance full-word barrett multiplier designed for FPGAs with DSP resources. In: Proceedings of the 15th conference Ph.D research in microelectronics and electronics (PRIME). 2. IEEE; 2019. p. 73–6. https://doi.org/10.1109/PRIME.2019.8787740.
[7] Parihar A, Nakhate S. Fast Montgomery modular multiplier for Rivest-Shamir-Adleman cryptosystem. IET Inf Secur 2019;13:231–8. https://doi.org/10.1049/iet-ifs.2018.5191.
[8] Liu W, Ni J, Liu Z, Liu C, O'Neill M. Optimized modular multiplication for supersingular isogeny diffie-hellman. IEEE Trans Comput 2019;68:1249–55. https://doi.org/10.1109/TC.2019.2899847.
[9] Kong Y, Hossain S. FPGA implementation of modular multiplier in residue number system. In: Proceedings of the IEEE international conference on Internet of Things and intelligence system; 2018. p. 137–40.
[10] Bos JW, Friedberger SJ. Faster modular arithmetic for isogeny-based crypto on embedded devices. J Cryptogr Eng 2019. https://doi.org/10.1007/s13389-019-00214-6.
[11] Brasilino LRB, Swany M. Low-latency CoAP processing in FPGA for the Internet of Things. In: Proceedings of the IEEE international conference on Internet of Things and IEEE green computing and communications and IEEE cyber, physical and social computing and IEEE smart data. IEEE; 2019. p. 1057–64. https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00182.
[12] Huang SZ, Chen RQ. FPGA-based IoT sensor HUB. In: Proceedings of the international conference on sensor networks and signal processing. IEEE; 2018. p. 139–44. https://doi.org/10.1109/SNSP.2018.00034.
[13] Song H, Fink GA, Jeschke S. Lightweight crypto and security. Cyber-physical systems: foundations, principles and applications. IEEE; 2017. p. 243–61. https://doi.org/10.1002/9781119226079.ch12.
[14] Schneier B. Applied cryptography. 2nd ed., 13. Indianapolis, Indiana: John Wiley & Sons, Inc.; 2015. https://doi.org/10.1002/9781119183471.
[15] Pratibha K, Muthaiah R. Survey on hardware implementation of montgomery modular. Int J Pure Appl Math 2018;119:13437–52.
[16] Javeed K, Irwin D, Wang X. Design and performance comparison of modular multipliers implemented on FPGA platform. Lecture notes in computer science, 10039. LNCS; 2016. p. 251–60. https://doi.org/10.1007/978-3-319-48671-0_23 (Including Subser Lect Notes Artif Intell Lect Notes Bioinformatics).
[17] Rodríguez-Flores L, Morales-Sandoval M, Cumplido R, Feregrino-Uribe C, Algredo-Badillo I. Compact FPGA hardware architecture for public key encryption in embedded devices. PLoS One 2018;13:1–21. https://doi.org/10.1371/journal.pone.0190939.
[18] Ghosh S, Mukhopadhyay D, Chowdhury DR. High speed Fp multipliers and adders on FPGA platform. In: Proceedings of the conference on design and architectures for signal and image processing. IEEE; 2010. p. 21–6. https://doi.org/10.1109/DASIP.2010.5706241.
[19] Javeed K, Wang X. Radix-4 and radix-8 booth encoded interleaved modular multipliers over general Fp. In: Proceedings of the 24th international conference on field programmable logic and applications (FPL); 2014. https://doi.org/10.1109/FPL.2014.6927452.
[20] Ding J, Li S. Broken-Karatsuba multiplication and its application to Montgomery modular multiplication. In: Proceedings of the 27th international conference on field programmable logic and applications (FPL); 2017. p. 5–8. https://doi.org/10.23919/FPL.2017.8056769.
[21] Chow GCT, Egurot K, Luk W, Leong P. A Karatsuba-based Montgomery multiplier. In: Proceedings of the international conference on field programmable logic and applications (FPL); 2010. p. 434–7. https://doi.org/10.1109/FPL.2010.89.
[22] Liu R, Li S. A design and implementation of Montgomery modular multiplier. In: Proceedings of the IEEE international symposium on circuits and systems (ISCAS). IEEE; 2019. p. 1–4. https://doi.org/10.1109/ISCAS.2019.8702684. vol. 2019- May.
[23] Yan X, Wu G, Wu D, Zheng F, Xie X. An implementation of Montgomery modular multiplication on FPGAs. In: Proceedings of the international conference on information science and cloud computing (ISCC); 2013. p. 32–8. https://doi.org/10.1109/ISCC.2013.19. 2014.
[24] Javeed K, Wang X, Scott M. Serial and parallel interleaved modular multipliers on FPGA platform. In: Proceedings of the 25th international conference on field programmable logic and applications (FPL); 2015. p. 2–5. https://doi.org/10.1109/FPL.2015.7293986.
[25] Liu R, Li S. A design and implementation of Montgomery modular multiplier. In: Proceedings of the IEEE international symposium on circuits and systems (ISCAS); 2019. p. 1–4. https://doi.org/10.1109/ISCAS.2019.8702684. vol. 2019- May.

Ahmed A. H. Abdelkader: He is a Telecom Egypt Managing- Director, Qussier, Egypt. He received the B.Sc. (Honors), M.Sc. , and Ph.D degrees in Electronics & Communications Engineering from Minia University, Egypt. His research interests include reconfigurable computing, FPGA, applied mathematics, computer science, cryptography, computer security.

Mostafa Rashdan: He received the B.Sc. and M.Sc. degrees in Electronics and Communications from Minia University, Egypt. He received the Ph.D. from the University of Calgary, Canada. He is an Assistant Professor in American University of the Middle East, Kuwait. His research interests include the mixed signal integrated circuit design, time-based serial communication link architectures, and high-speed data converters.

El-Sayed A. M. Hasaneen: He received his Ph.D. from the University of Connecticut, USA. He is currently a Professor in the Electrical Engineering Department, Faculty of Engineering at Aswan University, Egypt. His research interests are simulation of quantum dot non-volatile memories, solar cells, single electron transistors, and simulation of 15 nm mixed-signal circuits.

Hesham F. A. Hamed: He received the B.Sc., M.Sc., and Ph.D. degrees in Electronics and Communications Engineering from Minia University, Egypt. He is the Professor of Electronics, Faculty of Engineering, Minia University, and Egyptian-Russian University, Egypt. His current research interests include analog and mixed-mode circuit design, and digital integrated circuit design, and FPGA.