# A high speed Montgomery processor of 256-bit on FPGA

Xiaonan Zhang[1,a] and Xiuze Dong[2],

[1]*Institute of Communication Engineering, 710071, Xidian University, Xi'an, China*
[2]*Electronic Engineering Laboratory, 100070, Beijing Electronics Science & technology Institute, Beijing, China*

**Abstract.** To improve the speed of modular multiplication operation on ECC processor over GF (p), the paper presented a novel hardware implementation of Montgomery algorithm. Based on analyzing basic Montgomery modular multiplication algorithm, this work applied multi-step operation to Montgomery algorithm, which can accelerate speed by reducing the number of clocks. Simulation with Modelsim indicates that a completion of modular multiplication requires only 16 clock circles. Finally, the design of hardware architectures was evaluated on Altera Stratix III families. By following the new design concept, the multiplier structure can reach to a higher performance. Compared with other modular multiplier, the computation time of improved modular multiplier is lesser, decreasing 42% and reaching to 0.2 $\mu s$ . It is estimated that the computation of a 256-bit scalar point multiplication over GF(p) would take about 0.76 ms.

**Keywords:** ECC; Montgomery modular multiplication algorithm; multi-step operation.

## 1 Introduction

Modular multiplier is very important for public key cryptosystems, such as RSA [1] and ECC [2].However, compared with other public key cipher algorithm, ECC has more higher speed, smaller storage space and lower bandwidth requirements. Especially, due to limited computing resources, all kinds of wireless devices and smart cards are implemented on ECC. It is well-known that modular arithmetic is an important unit for the speed of ECC. So, it is particularly critical for ECC to accelerate the computation of modular multiplication.

Nowadays, there have been various algorithms literature proposed in speeding up modular multiplication. Among the modular multiplication schemes, their means can be broken down into two general types, namely, multiply and then reduce and Montgomery's method. The former means is efficiently performance when the modulo is a Merssane prime [3].In the previous work , they have proposed many effective methods. In M. Kaihara et al.[4] and M. Schramm et al. [5], they present shift and add modular multiplication algorithm; In prime field ECC processors, carry free structure is necessary to avoid lengthy data paths caused by carry propagation. There has been redundant schemes applied to different designs, for example, Carry Save Arithmetic (CSA) or Redundant Signed Digits (RSD); There is a scalable word based structure proposed in Tenca and Koc[6]. In D. Harris et al.[7], the authors present a scheme that used left shifted multiplication and modulus to replace intermediate result. In Rupali Verma et al.[9], this paper presents compute early word based scalable Montgomery architecture. It computes the most significant bit of word by applying 2 XOR operations. Recently, in

[a] Corresponding author : zhangxiaonan1010@163.com

 

2016 Shiann-Rong Kuang et al.[10], this paper proposes a simple and efficient Montgomery multiplication algorithm such that the low-cost and high-performance Montgomery modular multiplier can be implemented accordingly, which avoids the carry propagation at each addition operation. Hang Yuan et al[12] presented systolic array Montgomery multiplier. It was seen that the frequency of design is higher in 0.13 ASIC, but it occupied lots of clock numbers to calculate Montgomery modular multiplication.

Although the previous Montgomery algorithms are efficient, most of them occupied many DSP units to reduce computation time. So, most of previous works should not be flexible and well transportability. Therefore, this paper describes the design and implementation of a 256-bit Montgomery modular multiplication without occupying internal DSP units. The design is can applied to kinds of hardware platform such as ASIC, which has a good portability and flexibility. Section II presents concept and processor of ECC, and basic architecture of Montgomery modular multiplication. Improved structure of Montgomery modular multiplication is described section III. In section IV, hardware implementation, simulation results and comparisons with other designs are introduced. Finally, author makes a conclusion about this paper.

## 2 ECC and Montgomery modular multiplier

### 2.1 ECC structure

There are four floors to describe ECC structure, just as shown in Figure 1. The top floor should be the ECC protocol. The realization of protocol floor is determined by floor2 point multiplication, where computation of point multiplication can be implemented by floor3 point doubling and point addition. Basic operations in finite field are the lowest, including four parts, ie. Modular adder, modular subtraction, modular multiplication and modular inversion.

Modular adder and modular subtraction can be realized easily. Modular inverse is most time consuming operation, but point doubling and point addition can avoid modular inverse by transforming into Jacobi coordinates. Therefore, improving the speed of modular multiplication is initial for ECC scheme. This paper fosses on studying modular multiplication algorithm.

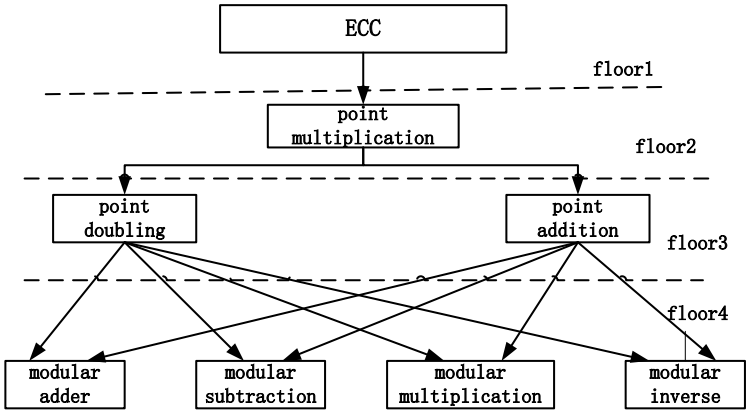


**Figure 1.** ECC Structure

### 2.2 Montgomery modular multiplier arithmetic

The algorithm for modular multiplication has been proposed by P.L. Montgomery in 1985[11].The algorithm is considered to be the fastest algorithm to compute AB mod M in computers when the values of A,B and M are large. It is a method for performing modular multiplication without division operation[12]. The following part gives a brief description of the Montgomery algorithm.

Suppose that using equation (1) calculates the modular multiplication :

$$z = xy \bmod m \tag{1}$$

Generally, people should calculate multiplication $x \bullet y$ ,then calculate reminder by division $(x \bullet y) / p$ .The procedure is more time-consuming, so Montgomery is put forward. It is well-known that the result of Montgomery product is the form :

$$xyR^{-1} \bmod m \tag{2}$$

Therefore, computation of modular multiplication can be solved by reasoning method to transform Montgomery product.

The basic idea of Montgomery modular multiplication algorithm is calculation of $xyR^{-1} \bmod m$ . Consider two natural numbers m and $R > m$ , where m is a k-bit number. And assume that they are relative prime, that is, $\gcd(m, R) = 1$ . Generally speaking, $R = 2^k$ . Then there exists an element $R^{-1}$ of $Z_m$ such that $RR^{-1} \bmod m = 1$ ; The relationship of R and m is $RR^{-1} - mm' = 1$ ,where $m' \equiv -m^{-1} \bmod R$ . Algorithm1 gives Montgomery multiplication of two given numbers x and y , and the result is : $z = xyR^{-1} \bmod m$ .

Algorithm 1 Montgomery multiplication algorithm structure

Input : x, y, m, R,

Output : $z \leftarrow xyR^{-1} \bmod m$

Step1 : $A \leftarrow x \bullet y$ ;

Step2 : $B \leftarrow A \bmod R$ ;

Step3 : $\lambda \leftarrow B \bullet m' \bmod R \ \ (0 \le \lambda \le R)$ ;

Step4 : $z \leftarrow (A + \lambda m) / R$ ;

Step5 :if $z \ge m$

then $z \leftarrow z - m$ ;

else

return $z \leftarrow z$

end if

## 3 Improved Montgomery modular multiplier

According to Algorithm1, Montgomery ‘s original algorithm need to do $k \times k$ -bit multiplication and $2k$ -bit addition. What’s more, its middle results reach to 2n-bit, so the consumption of hardware resources are incalculable and not implemented on hardware. Therefore, Algorithm1 needs optimizing appropriately such that Algorithm1 is more suitable for hardware implementation.

### 3.1 Hardware structure of Montgomery

If m is odd then there exists an element $2^{-1}$ of $Z_m$ such that $2 \bullet 2^{-1} \bmod m \equiv 1$ .

Assume m is k-bit number, $R = 2^k$ , $x = x_{k-1} \bullet 2^{k-1} + x_{k-2} \bullet 2^{k-2} + \cdots + x_0 \bullet x^0$

So,

$$
\begin{aligned}
M(x,y) &= xyR^{-1} \bmod m = (x_{k-1}2^{k-1} + x_{k-2}2^{k-2} + \cdots \\
&\quad + x_0 2^0) y (2^k)^{-1} \bmod m \\
&= (x_{k-1} y 2^{k-1} + x_{k-2} y 2^{k-2} + \cdots + x_0 y 2^0)(2^k)^{-1} \bmod m \\
&= ((\cdots(((0 + x_0 y)2^{-1} + x_1 y)2^{-1} + x_2 y)2^{-1} + \\
&\quad \cdots)2^{-1} + x_{k-1} y)2^{-1} \bmod m
\end{aligned} \tag{3}
$$

If a natural $a$ is multiplied by $2^{-1}$ ,the result can be expressed by following form :

$$
\begin{cases}
a2 \equiv a/2 \bmod m & if \ a \bmod 2 = 0 \\
a2 \equiv (a+m)/2 \bmod m
\end{cases} \tag{4}
$$

Therefore ,Algorithm2 is more suitable for hardware implementation.

Algorithm2 :Hardware implementation of Montgomery modular multiplication

Input : $x, y, m, k, R$

Output : $z = xyR^{-1} \bmod m$

Step1 :Assume $p \leftarrow 0$ ;

Step2 : for $i = 0$ to $k-1$, do

begin

$a \leftarrow p + x[i] \bullet y$ ;

if ( $a \bmod 2 = 0$ )

then $p \leftarrow a/2$ ;

else

$p \leftarrow (a+m)/2$ ;

end if

end

end for

Step3 :if $(p \geq m)$

$z \leftarrow p - m$

else $z \leftarrow p$

end if

### 3.2 Optimization of hardware implementation

Generally, structure of the hardware implementation can be divided into the serial structure, parallel structure and string and hybrid structure. Full parallel structure need to complete all operations in one clock cycle. The means can reduce time largely, but it completes a calculation at the expense of the large hardware resources. Therefore, the means have only theoretical research significance. Full serial structure is that only one iteration computation is completed in each clock cycle ,so it needs a lot of clock cycles. Compared with full parallel structure, this form uses less hardware resources, but speed is quite slow. As a result, serial-parallel structure can be applied to the Montgomery algorithm.

Compared with the original algorithm, our improved algorithm is more flexible. Firstly, we introduced the three variables s and t and r, respectively, represents the steps in a clock, steps execution times, and the rest of the three concepts, calculation of two exactly getting originally should have need of Montgomery reduced to only t k a clock a clock or t + 1 clock (one). In addition, the relation between the three variables:

$$k = s \bullet t + r \tag{5}$$

Algorithm3 shows the proposed multi-step structure. It describes how to realize the relationship multi-step and reducer. In order to more clearly illustrate the method of proposed implementation, the author also gives Fig 3 to explain the structure of hardware implementation.

Algorithm3: Multi-step Hardware Implementation

Input: $x, y, s, t, r, m, k, R$

Output: $z = xyR^{-1} \bmod m$

Step1:Assume $p \leftarrow 0$ ;

Step2: for $i = 0$ to $t - 1$, do
for $i = 0$ to $k - 1$, do
begin
$a \leftarrow p + x[i] \bullet y$ ;
if $a \bmod 2 = 0$
then $p \leftarrow a / 2$ ;
else
$p \leftarrow (a + m) / 2$ ;
end if
end
end for
end for

Step3:if $(r > 0)$
for $i = 0$ to $k - 1$, do
begin
$a \leftarrow p + x[i] \bullet y$ ;
if $a \bmod 2 = 0$
then $p \leftarrow a / 2$ ;
else
$p \leftarrow (a + m) / 2$ ;
end if
end
else
turn Step 4;
end if

Step4: if $p \geq m$
$z \leftarrow p - m$
else $z \leftarrow p$
end if

Figure 2and Figure 3 shows the multi-step Montgomery structure, where Figure 2 represents Architecture of multi-step Montgomery for hardware implementation. From Figure2, we can see that control unit and CSA_Montgomery unit are most important parts. the structure consists of two parts, ie. Main Montgomery structure and Remainder Montgomery structure, respectively, iterations in a clock and remainder steps. For Montgomery structure in Figure2, it is showed in Figure 3.

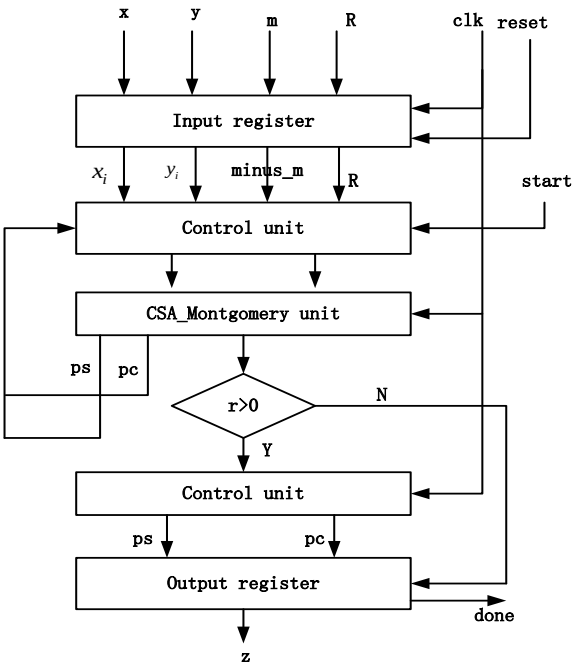


**Figure 2.** Architecture of multi-step Montgomery for hardware implementation

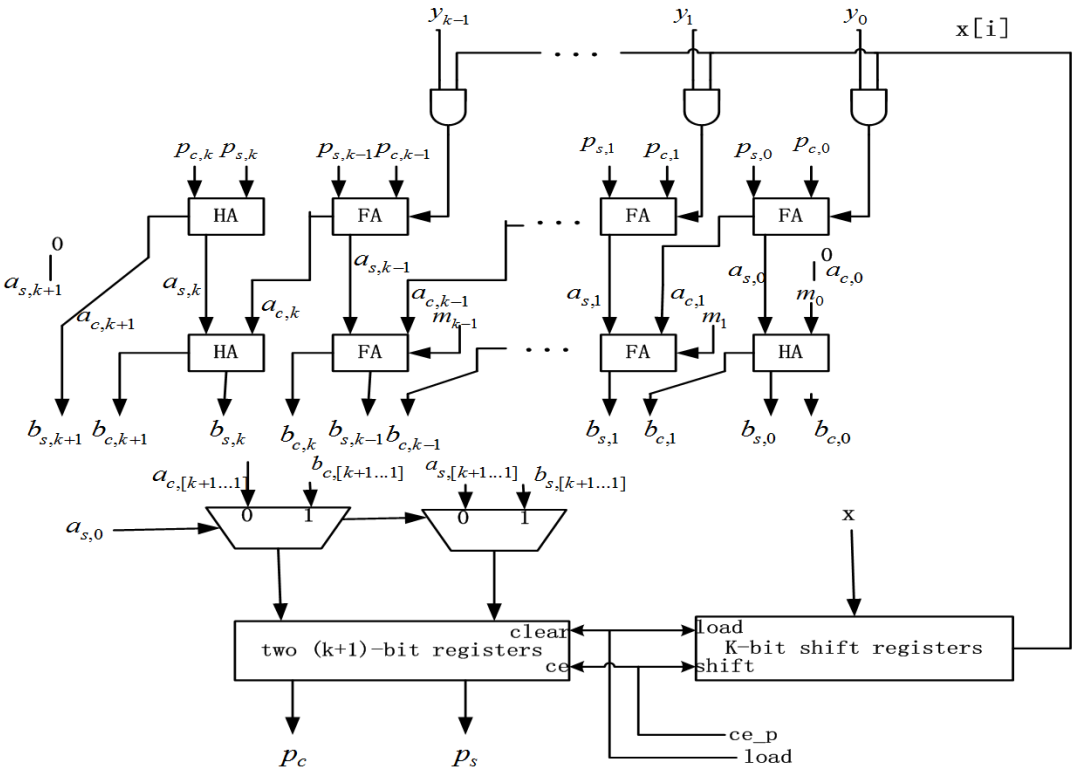


**Figure 3.** the structure of CSA_Montgomery unit

## 4 Result and performance comparison

In order to better analyze the Montgomery , the paper presents the results of new architecture. The correctness of design is verified by Modelsim . With the correctness, firstly, we select Altera Stratix III as the target device, then the design is developed in Verilog HDL and synthesized with Quartus II 13.0. Figure 4 gives the circuit structure from RTL view.

The improved design implemented on Altera Stratix III takes 17161 ALUTs, ,about 37% of all resources in the device, and .no using DSP units. The reset resource is enough for other IPs to build a full cryptographic system.

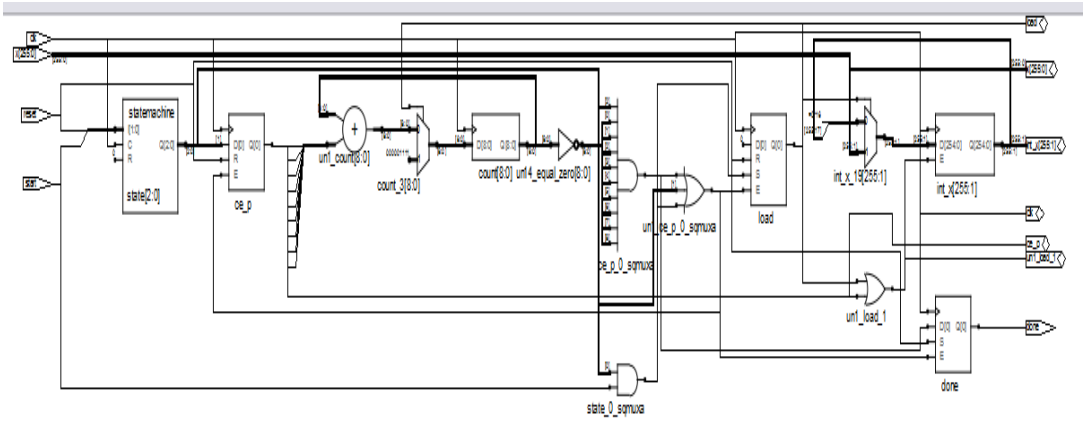


**Figure 4.** RTL view for multi-step Montgomery modular multiplication

The performance comparison with previous relevant designs is illustrated in Table1. Hang Yuan et al[12] presented systolic array Montgomery multiplier. We can see that the frequency of design is higher in 0.13 ASIC, but it occupied lots of clock numbers to calculate Montgomery modular multiplication. Literature [13] and literature [14] used embedded multipliers structure. Former method was 256 embedded multipliers, and latter idea was 81 embedded multipliers. Both of them used lesser clock numbers relative to literature [12] such that cost lesser time, but they occupied DSP units expect relative resources, where literature [13] used 256 $18\times18$ mult and 11992 slices, literature [14] has taken 81 embedded multipliers and 23405 LES. Therefore, both of designs are not better than ours in balancing speed and hardware resources. Fully- parallel that presented by Mr. Nahed Mulla et al[15] is similar to our design. Compared with Mr. Nahed Mulla et al[15], our dsign may be cost more time, but their design occupied much more resources, including 23405LE and 16DSP. So, their design did not have generic , that is to say, their design was not apply to other platform, such as ASIC. M. Morales-Sandoval et al[16] presented iterative digit–digit Montgomery multiplication (IDDMM) algorithm, and their input operands (multiplicand, multiplier and modulus) are represented using as radix $\beta = 2^k$ .It can be seen that the performance of k=64 is more efficient than the performance of k=32.However, the method of literature[16] also occupied many DSP units, so our design is quite dominant in trade-off speed and hardware.

**Table 1.** A comparison of Montgomery multiplication for hardware implementation

| design | platform | Montgomery algorithm | Frequency (MHz) | Clock numbers | Time ( $\mu s$ ) |
|---|---|---|---|---|---|
| [12] | 0.13 ASIC | Systolic array | 100 | 120 | 1.2 |
| [13] | Xilinx Virtex2 | 256 embedded multipliers used | 45.6 | 32 | 0.7 |
| [14] | Altera Cyclone3 | 81embedded multipliers used | 30.38 | 15 | 0.49 |
| [15] | Altera Cyclone3 | fully-parallel | 30.05 | 3 | 0.1 |
| [16] | Altera Spartan3 | proposed IDDMM(k=32) | 63.8 | 79 | 1.25 |
| [16] | Xilinx Virtex5 | proposed IDDMM(k=64) | 68.24 | 23 | 0.35 |
| This work | Altera Stratix III | Serial-parallel | 76.8 | 16 | 0.2 |

## 5 Conclusion

In this paper, we put forward a kind of more effective multi-step Montgomery multiplication algorithm based on multi-step method. The proposed architecture can realize the 256-bit Montgomery multiplication calculation of arbitrary steps. With the 16 steps, it can perform a modular multiplication in only 16 cycles. Results on Altera StratixIII shows that the paper's design is more faster than previous designs on ASIC or FPGA devices. Compared with previous relevant works, the design not only has higher speed, but also occupies lesser hardware resources. What's more, the design can also be performed other platform due to using internal DSP units. In order to explain preferablely the importance of proposed Montgomery for ECC, we would accomplish an entire ECC structure to estimate advantage of the Montgomery design in the future.

## Acknowledgement

This work is supported by Natural Science Foundation of Beijing.(Grant No.4163076).

## References

1. Rivest, R.L., Shamir, A., Adleman, L.: 'A method for obtaining digital signatures and public-key cryptosystems', Commun. ACM, **21**, 2, pp. 120–126,(1978).
2. Miller, V.: 'Use of elliptic curves in cryptography'. Proc. of Advances in Cryptology, CRYPTO'85, Santa Barbara, CA, pp. 417–426, August (1985).
3. J. Solinas. "Generalized Mersanne Number," Technicial Report CORR. 39-99(1999).
4. M. Kaihara and N. Takagi, "A Hardware algorithm for modular multiplication/division based on the extended Euclidean algorithm", IEICE Trans. Fundamentals, E88-A, **12**, pp.3610-3617, Dec.(2005).
5. M. Schramm and A. Grzemba , "On the implementation of a lightweight generic FPGA ECC crypto-core over GF(p)", 2013 International Conference on Applied Electronics(AE), pp.1-4,(2013).

6. A.F. Tenca, and C.K. Koc, “A Scalable Architecture for Modular Multiplication Based on Montgomery’s Algorithm,” IEEE Trans. Computers, vol. **52**, no. 9, pp. 1215-1221, Sept. (2003).
7. D. Harris, R. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu, “An Improved Unified Scalable Radix-2 Montgomery Multiplier,” Proc. 17th.IEEE Symp. Computer Arithmetic (ARITH), pp. 172-178, June (2005).
8. D.S. Chen, H.T. Li, and Y.W. Wang, ‘‘A Prediction-Based Scalable Design for Montgomery Modular Multiplication,” Proc. 3 rd International Conference on Electric and Electronics, pp. 46-50, (2013).
9. Rupali Verma, Maitreyee Dutta and Renu Vig, “Early-Word-Based Montgomery Modular Multiplication Algorithm,” 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), pp. 595-600,(2015).
10. Shiann-Rong Kuang, Kun-Yi Wu, and Ren-Yao Lu, “Low-Cost High-Performance VLSI Architecture for Montgomery Modular Multiplication,” IEEE Transactions on very large scale integration (VLSI) systems, **24**, 2, pp.432-443, (2016).
11. P. Montgomery, “Modular multiplication without trial division,” Mathematics of computation, **44**, 170, pp. 519–521, (1985).
12. Hang Yuan, Platform Based Soc Design Study of Elliptic Curve Cryptogarphy (in Chinese). Dissertation of Tsinghua University, (2005)
13. C.J.McIvor, M.McLoone, and J.V.McCanny,”Hardware Elliptic Curve Cryptographic Processor Over GF(P), IEEE Transacitons on circuits and systems-I:Regular Papers. **53**, 9, September (2006).
14. Y. Gong and S. Li, ”High-throughput FPGA implementation of 256-bit Montgomery modular multiplier”, 2010 Second International Workshop on Education Technology and Compute Science(ETCS), pp.173-176, (2010).
15. Mr. Nahed Mulla and Mr. Abhay Kasetwar, “FPGA Implementation of an Efficient Montgomery Multiplier For Adaptive Filtering Application”, 2014 IEEE, pp.66-70,(2014).
16. M. Morales-Sandoval and A. Diaz-Perez.”Scalable GF(p) Montgomery multiplier based on a digit–digit computation approach”, IET Computers & Digital Techniques, **10**, 3, pp. 102–109, (2016).