Fast Modular Multiplication Execution in Residue Number System

N. Chervyakov[#1], M. Babenko[#2], A. Tchernykh[*3], V. Kuchukov[#4], M. Deryabin[#5], N. Kuchukova[#6]

[#]*Mathematics and Mathematical Modelling, North-Caucasus Federal University*
Stavropol, 355009, Russian Federation;
[*]*CICESE Research Center*
Ensenada, Baja California, Mexico
[1]k-fmf-primath@stavsu.ru, [2]mgbabenko@ncfu.ru, [3]chernykh@cicese.mx, [4]viktor-kuchukov@yandex.ru,
[5]maderiabin@ncfu.ru, [6]knn.storage@yandex.ru

***Abstract* — In the paper, we propose a new method of modular multiplication computation, based on Residue Number System. We use an approximate method to find the approximate method a residue from division of a multiplication on the given module. We substitute expensive modular operations, by fast bit right shift operations and taking low bits. The carried-out simulation on Kintex7 XC7K70T board showed that the offered method allows to win in time on average for 75%, and in the area -- on average for 80% relatively to modified method from work [1] that makes it more applicable for the hardware implementation of the cryptography primitives constructed over a simple finite field.**

Keywords— ***approximate method; FPGA; Montgomery algorithm; residue number system*.**

I. INTRODUCTION

Finding remainder of division of a number by the fixed module is basic operation of a large number of algorithms implementation: information security ([1]-[5]), digital signal processing [6], wireless systems [7], etc. During development of the hardware decisions for the modern information systems special attention is paid to technical characteristics: speeds of operation, area, etc. Use of the Residue Number System (RNS) allows to execute addition and multiplication of numbers on parallel computing channels without bit carrying between channels that allows to increase the speed of arithmetical operations execution. In the paper a new method of modular multiplication computation, based on finding by the approximate method a residue of division of a multiplication by the given module in Residue Number System (RNS) is proposed. Due to use of approximate method from work [8] for finding a remainder of division expensive modular operations which are replaced with fast bit right shift operations and taking low bits aren't required.

II. RESIDUE NUMBER SYSTEM

The theory of congruence relation and Chinese Remainder Theorem are the cornerstone of RNS. Arithmetical operations (addition, multiplication) in RNS are executed parallely independently on L channels and without carrying between computing channels. It's worth noticing that in each separate computing RNS channel we carry out operation with numbers of smaller size – with number remainders of division on the RNS module that leadds to reduction of carries number and reliability and the speed of arithmetical operations with numbers execution augmentation. RNS is defined by pairwise coprime numbers $m_1, m_2, ..., m_L$, called modules. RNS range can be computed by the formula $M = \prod_{i=1}^{L} m_i$. Any integer A belonging to a segment $[0, M-1]$ is represented unambiguously in RNS with a tuple $(a_1, a_2, ..., a_L)$ where for all $i = \overline{1,n}$ congruence $a_i = A \bmod m_i$ is true.

According to Chinese Remainder Theorem number A can be recovered with the use of formula:

$$A = \left| \sum_{i=1}^{L} \frac{M}{m_i} \left| M_i^{-1} \right|_{m_i} a_i \right|_M , \qquad (1)$$

where $M_i = M / m_i$ and $\left| M_i^{-1} \right|_{m_i}$ - multiplicative inverse of M_i modulo m_i.

If we divide (1) by constant M, then we will get the approximate value

$$\frac{A}{M} = \left| \sum_{i=1}^{L} \frac{\left| M_i^{-1} \right|_{m_i}}{m_i} a_i \right|_1 = \left| \sum_{i=1}^{L} k_i a_i \right|_1 , \qquad (2)$$

where $k_i = \dfrac{\left| M_i^{-1} \right|_{m_i}}{m_i}$ - constants of the chosen system and a_i - remainders of number A represented in RNS, thus value of expression (2) will be in an interval $[0,1)$. The end result of the amount is computed after summing and discarding of an integer part of number with saving a fractional part of the amount. The fractional part can be also written as $X \bmod 1$, because $X = \lfloor X \rfloor + X \bmod 1$.

Follows from a formula (2) that the RNS to positional notation conversion is made by a formula:

978-1-5090-3680-6/16/$31.00 ©2016 IEEE

$$A = M\left|\sum_{i=1}^{L} k_i a_i\right|_1 \qquad (3)$$

III. The Review of Works of Modular Multiplication Computation in RNS

Algorithms of modular multiplication computation can be divided into two classes: the ones, which use an operation of finding the remainder of division by the fixed module of a product and the others, which use the operations of finding residual during multiplication computation process. We will consider the algorithms which allow executing modular multiplication without use of operation of finding the remainder from division in general but require precomputation and storage of constants.

The Montgomery algorithm for modular multiplication without division is proposed in work [9]. The effective systolic realization of the Montgomery algorithm is proposed in work [2] and is presented below by the algorithm:

Algorithm 1. MMM (A_R, B_R, p)

1. If $(b_0 = 1)$ then $B_R = B_R - 1$; $T^{(0)} = A_R$;
2. For $i = 0$ to $n+1$
2.1. For $j = 0$ to $n+2$
2.1.1. $t' = (t_{i,j} \oplus C0_{i,j}) \oplus (a_i \wedge b_j)$;
2.1.2. $C0_{i,j+1} = ((t_{i,j} \oplus C0_{i,j}) \wedge (a_i \wedge b_j)) \vee (t_{i,j} \wedge C0_{i,j})$;
2.1.3. $t_{i+1,j+1} = t' \oplus ((t_{i,0} \wedge n_j) \oplus (D0_{i,j} \vee D1_{i,j}))$;
2.1.4. $D0_{i,j+1} = t' \wedge ((t_{i,0} \wedge n_j) \oplus (D0_{i,j} \vee D1_{i,j}))$;
2.1.5. $D1_{i,j+1} = (t_{i,0} \wedge n_j) \wedge (D0_{i,j} \vee D1_{i,j})$;

where $p-$ n-bit odd module, $R = 2^{n+2}$, $A_R = A \cdot R \bmod p$, $B_R = B \cdot R \bmod p$ and the result $T = [t_{n+2,n}, \ldots, t_{n+2,0}]$ satisfies a condition $A_R, B_R, T \in [0, 2p)$; in beginning of cycle $T^{(0)}$, $C0$, $D0$ and $D1$ are zero. In work [3] it is said that critical way of this algorithm delay is $3T_{XOR}$, where T_{XOR} delay of two input element XOR.

The modular multiplication algorithm of Montgomery in two fields and generalization of set of modules conversion algorithm for two fields is shown in work [10]. The multiplication algorithm where $\oplus$ multiplication/subtraction operation in two fields and $\otimes$ - multiplication operation in two fields is set as:

Algorithm 2. RMM $(A_\tau, B_\tau, (-p^{-1})_\beta, Q_\alpha^{-1}, p_\alpha)$

1. $s_\tau = A_\tau \otimes B_\tau$;
2. $c_\beta = s_\beta \otimes \left(-p^{-1}\right)_\beta$;
3. $c_\alpha = c_\beta$ (set of modules conversion);
4. $u_\alpha = c_\alpha \otimes p_\alpha$;
5. $v_\alpha = s_\alpha \oplus u_\alpha$;
6. $T_\alpha = v_\alpha \otimes Q_\alpha^{-1}$;
7. $T_\beta = T_\alpha$ (set of modules conversion).

In this algorithm sets of modules are $\alpha = (p_1, p_2, \ldots, p_L)$ and $\beta = (q_1, q_2, \ldots, q_L)$ such that $\gcd(p_i, q_j) = 1$, $\forall i, j \in [1, L]$. Input are numbers A and B represented by two sets of RNS modules, i.e. A_τ and B_τ, constant $(-p^{-1})_\beta$ in RNS β, constants Q_α^{-1} and p_α in RNS α, where $A, B < 2p$, $Q = \prod_{i=1}^{L} q_i$. The output of this algorithm is T_τ, where $T < 2p$ and $T \equiv A \cdot B \cdot Q^{-1} \bmod p$. During execution of multiplication it is necessary to transfer between systems of the bases α and β twice.

Important aspect of the given algorithm is that it comes down to simple accumulative multiplication where the word length is equal to module length. It allows to construct the device with completely parallel architecture where each module corresponds to one modules of the RNS bases system.

IV. Effective Implementation of a Remainder of Division Computation In Rns

We will consider approach when computation of modular multiplication comes down to two stages: to find two numbers product $C = A \times B$ and to find residue of number modulo p, i.e. $C \bmod p$.

Effective implementation of finding number C remainder of division modulo p in RNS can be achieved with the formula (2):

$$\begin{aligned} \frac{C}{M} &= \left| p\sum_{i=1}^{L} \frac{\left|M_i^{-1}\right|_{m_i}}{p \cdot m_i} c_i \right|_1 = \left| p\sum_{i=1}^{L} \overline{\overline{k_i}} c_i \right|_1 \\ &= \left| \sum_{i=1}^{L} k_i c_i \right|_1 = p\sum_{i=1}^{L} \overline{\overline{k_i}} c_i - \left[\sum_{i=1}^{L} k_i c_i \right], \end{aligned} \qquad (4)$$

where $\overline{\overline{k_i}} = \dfrac{\left|M_i^{-1}\right|_{m_i}}{p \cdot m_i}$ and $[x]$ – denotes integer part of number x. From (4) follows that C can be represented:

$$C = \left(p\sum_{i=1}^{L} \overline{\overline{k_i}} c_i - \left[\sum_{i=1}^{L} k_i c_i \right] \right) \cdot M \qquad (5)$$

With the formula (5) we calculate value C / p:

$$\frac{C}{p} = \left(\sum_{i=1}^{L} \overline{\overline{k_i}} c_i - \frac{1}{p} \cdot \left[\sum_{i=1}^{L} k_i c_i \right] \right) \cdot M \qquad (6)$$

Therefore, value $C \bmod p$ can be calculated by a formula:

$$C \bmod p = C - \left\lfloor \frac{C}{p} \right\rfloor \cdot M =$$

$$= C - \left\lfloor \left(\sum_{i=1}^{L} \overline{\overline{k_i}} c_i - \frac{1}{p} \cdot \left[\sum_{i=1}^{L} k_i c_i \right] \right) \cdot M \right\rfloor \cdot p \quad (7)$$

As there is no need to make computation with integer parts of real numbers it is possible to make transition from computation with fractional parts to integer calculations. It can be made as follows:

- to multiply each real constant by 2^N, where N - the number of binary digits of the fractional part providing the necessary level of accuracy of computation;
- to ceil each received number;
- to make all computation in residue class ring over the module 2^N.

Using an assessment of N from work [9], we will receive that $N = \left\lceil \log_2 \left(M \cdot \sum_{i=1}^{L} (m_i - 1) \right) \right\rceil$. Considering the fact that in FPGA operation with real numbers is expensive, we will transfer from real numbers to integer having multiplied $\overline{\overline{k_i}}$ and k_i by 2^N, then the formula (7) will assume:

$$C \bmod p = C_{RNS} - K \cdot p_{RNS}, \quad (8)$$

where $K = \left\lfloor \left(\sum_{i=1}^{L} \widetilde{k}_i c_i - \mu \cdot \left\lfloor \sum_{i=1}^{L} \overline{k}_i c_i / 2^N \right\rfloor \right) \cdot \frac{M}{2^N} \right\rfloor$,

$\widetilde{k}_i = \left\lfloor 2^N \cdot \overline{\overline{k_i}} \right\rfloor$, $\overline{k}_i = \left\lfloor 2^N k_i \right\rfloor$, $\mu = \left\lfloor 2^N / p \right\rfloor$ and $N = \left\lceil \log_2 \left(M \cdot \sum_{i=1}^{L} (m_i - 1) \right) \right\rceil$.

As a result of the computation of $T = C \bmod p$ by the formula (8) algorithm work, a result satisfying a condition: $0 \le T < 2p$ will be received, analogically to Montgomery algorithm from works ([1]-[3], [9], [10], [11]).

Simulation is made on board Kintex7 XC7K70T.

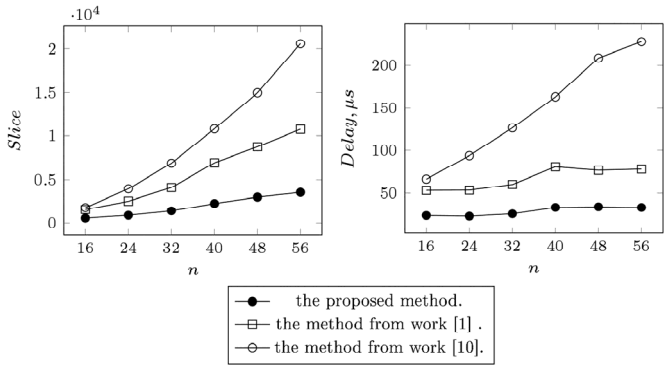

Fig. 1 Technical characteristics of FPGA modular multiplication architectures

From Fig. 1 it is obvious that the the offered method allows to win in time on average for 75%, and in the area on average for 80% relatively to modified method from work [1].

V. Conclusions

In the paper a new method of modular multiplication computation, based on finding by the approximate method a residue from division of a multiplication by the given module in RNS is proposed. Use of the approximate method for finding a remainder of division doesn't require expensive modular operations which are replaced with fast bit right shift operations and taking low bits. The carried-out simulation on Kintex7 XC7K70T board showed that the offered method allows to win in time on average for 75%, and in the area on average for 80% relatively to modified method from work [1] that makes it more applicable for the hardware implementation of the cryptography primitives constructed over a simple finite field.

Acknowledgment

This work is partially supported by the State task No. 2563 and Russian Federation President Grant SP-1215.2016.5.

References

[1] S.-H. Choi, and K.-J. Lee, "New systolic modular multiplication architecture for efficient Montgomery multiplication," *IEICE Electron. Express*, vol. 12, pp. 20141051-20141051, 2015.

[2] K. Manochehri, B. Sadeghian, and S. Pourmozafari, "A modified radix-2 Montgomery modular multiplication with new recoding method," *IEICE Electron. Express*, vol.7, pp. 513-519, 2010.

[3] S.-H. Choi, and K.-J. Lee, "Enhancement of a modified radix-2 Montgomery modular multiplication," *IEICE Electron. Express*, vol.11, pp. 20140782-20140782, 2014.

[4] C. D. Walter, "Montgomery exponentiation needs no final subtractions," *Electron. Lett.*, vol. 35, pp. 1831-1832, 1999.

[5] C. McIvor, M. McLoone, and J. McCanny, "Improved Montgomery modular inverse algorithm," *Electron. Lett.*, vol. 40, pp. 1110-1112, 2004.

[6] D. Zivaljevi'c, N. Stamenkovi'c, and V. Stojanovi'c, "Digital filter implementation based on the RNS with diminished-1 encoded channel," *TSP*, pp. 662-666, 2012.

[7] V. Yatskiv, S. Jun, N. Yatskiv, A. Sachenko, and O. Osolinskiy, "Multilevel method of data coding in WSN," *IDAACS*, pp. 863-866, 2011.

[8] N. Chervyakov, M. Babenko, P. Lyakhov, and I. Lavrinenko, "An approximate method for comparing modular numbers and its application to the division of numbers in residue number systems*," *Cybernetics and Systems Analysis*, vol. 50, pp. 977-984, 2014.

[9] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, pp. 519-521, 1985.

[10] D. Schinianakis, and T. Stouraitis, "Multifunction residue architectures for cryptography," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, pp. 1156-1169, 2014.

[11] D. Schinianakis, and T. Stouraitis, "A RNS Montgomery multiplication architecture," *ISCAS*, pp. 1167-1170, 2011.

.