Aravind Kundu* et al.
(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH
Volume No.5, Issue No.1, December – January 2017, 5560-5563.

Embedding Residue Arithmetic Into Modular Multiplication For Integers And Polynomials

Dr.ARVIND KUNDU
Associate Professor(HOD), Department of ECE
SCIENT Institute of Technology, Ibrahimpatnam, T.S, INDIA

Mr.V.NAGA MAHESH M.Tech
Assistant Professor, Department of ECE
SCIENT Institute of Technology, Ibrahimpatnam, T.S, INDIA

R.DEEPTHI
PG Scholar, Department of ECE
SCIENT Institute of Technology, Ibrahimpatnam, T.S, INDIA

Abstract: **A brand new methodology for embedding residue arithmetic inside a dual-field Montgomery modular multiplication formula for integers in as well as for polynomials was presented within this project. A design methodology for incorporating Residue Number System (RNS) and Polynomial Residue Number System (PRNS) in Montgomery modular multiplication in GF (p) or GF (2n) correspondingly, in addition to VLSI architecture of the dual-field residue arithmetic Montgomery multiplier are presented within this paper. In cryptographic applications to engender the public and private keys we suffer from the arithmetic operations like advisement, subtraction and multiplication. An analysis of input/output conversions to/from residue representation, combined with the suggested residue Montgomery multiplication formula, reveals prevalent multiply-accumulate data pathways both between your converters and backward and forward residue representations.**

Keywords: **Carry Save Addition; Energy Efficient Architectures; Montgomery Modular Multiplier; Cryptosystem;**

I. INTRODUCTION

The computation from the Montgomery exponentiation (ME) within the Residue Number System (RNS) sanctions constraining the delay because of carry propagation and reaching a higher amount of parallelism. This method mainly necessitates the execution of some Montgomery multiplications (MMs). However, in RNS, some operations are natively arduous to complete. Hence, several approaches happen to be suggested to be able to planarity exploit the potential for RNS for modular exponentiation, by minimizing the outcome of cognate drawbacks. A vital component of these approaches may be the Base Extension (BE), which calculates several on the different RNS base. Using residue systems can provide reduced complexity and power use of arithmetic units with large word lengths [1]. However, RNS/PRNS implementations bear the additional price of input converters to translate figures from the standard binary format into residues and output converters to translate from RNS/PRNS to binary representations. The mathematical conditions that should be satisfied for any valid RNS/PRNS incorporation are examined. The derived architecture is extremely parallelizable and versatile, because it supports binary-to RNS/PRNS and RNS/PRNS-to-binary conversions, Mixed Radix Conversion (MRC) for integers and polynomials, dual-field Montgomery multiplication, and dual-field modular exponentiation and inversion within the same hardware.

II. RNS TECHNIQUE

A residue number system is characterized by a predicate that is not a single radix but an N-tuples of integers (mN,mN-1 … m1). Each of these mi (i = 1, 2, … N) is called a modulus. An integer "X" is represented in the residue number system by N-tuple (xN, xN-1… x1) where xI is a nonnegative integer gratifying

$$X = m_I * q_I + x_I \quad , \ldots\ldots\ldots\ldots (1)$$

Where qI is the most sizably voluminous integer such that 0<=xI <= (mI – 1). xi is kenned as the residue of X modulo mi, and notations X mod mi and |X|mi are commonly utilized. In Residue Number Systems (RNS), an integer X is represented by its residues {x0…..xn-1} modulo a base of relatively prime numbers {m0…..mn-1}. Thus an astronomically immense number can be represented as a set of diminutive integers. Additament and multiplication can be facilely parallelized, there is no carry propagation. The time is reduced to the evaluation of these operations with diminutive numbers. This representation is utilizable in cryptography and digital signal processing. Furthermore, in these two domains, modular multiplication (A X B mod N) is frequently utilized. When a number gets more immensely colossal, the arithmetic operations get more involutes. Thus in RNS system, the sizably voluminous numbers can be represented into their residues. The residues are conspicuously minutely diminutive in comparison to the immensely colossal number [2]. Hence, the arithmetic

operations will be simple and can be performed on each residue independently giving the provision of parallel operations. The RNS divides an integer into a number of more diminutive integers that can be processed in parallel independently of each other.

III. RSA ALGORTIHM

RSA implemented two important ideas: i) Public-key file encryption. This concept omits the requirement for a "courier" to provide secrets of recipients over another secure funnel before transmitting the initially-intended message. In RSA, file encryption keys are public, as the understanding keys aren't, so just the person using the correct understanding key can decipher an encrypted message. Everybody has their very own file encryption and understanding keys. The keys should be made in a way the understanding key might not be easily deduced in the public file encryption key. ii) Digital signatures. The receiver might need to verify that the transmitted message really originated in the sender (signature), and didn't just originate from there (authentication).

IV. MONTGOMERY MULTIPLICA-TION

Let the modules n be a k-bit integer i.e., $2^{k-1} \leq n < 2^k$ and let r be 2^k.The Montgomery Multiplication algorithm requires that r and n be relatively prime, i.e., gcd(r,n)=gcd(2^k,n)=1.this requirement is satisfied if n is odd. in order to describe the Montgomery multiplication algorithm, we first define the n-residue of an integer a<n as a=a.r(mod n).it is the straightforward to show the set is a complete residue system. {a.r mod n ,0 ≤ a ≤ n-1} Montgomery multiplication requires converting a and b into a special representation called Montgomery form. However, when many products are required, as in modular exponentiation, the conversion to Montgomery form becomes a negligible fraction of the time of the computation, and performing the computation by Montgomery multiplication is faster than the available alternatives. Let N denote a positive integer modulus. The ring Z/NZ consists of residue classes modulo N, that is, sets of the form:

$$\{a + kN : k \in \mathbf{Z}\},$$

where a is some fixed integer. Each residue class is a set of integers such that the difference of any two integers in the set is divisible by N [3]. The residue class corresponding to a is denoted $\bar{a}$. Equality of residue classes is called congruence and is denoted:

$$\bar{a} \equiv \bar{b} \pmod{N}.$$

Storing an entire residue class on a computer is impossible because the residue class has many elements. Instead, residue classes are stored as representatives. Conventionally, these representatives are the integers a for which $0 \leq a \leq N - 1$. If a is an integer, then the representative of $\bar{a}$ is written a mod N. When writing congruence's, it's common to identify an integer with the residue class it represents. With this convention, the above equality is written

$$a \equiv b \pmod{N}$$

Arithmetic on residue classes is done by first performing integer arithmetic on their representatives. The output of the integer operation determines a residue class, and the output of the modular operation is determined by computing the residue class's representative. For example, if N = 17, then the sum of the residue classes $\overline{7}$ and $\overline{15}$ is computed by finding the integer sum 7 + 15 = 22, then determining 22 mod 17, the integer between 0 and 16 whose difference with 22 is a multiple of 17. In this case, that integer is 5, so

$$\overline{7} + \overline{15} \equiv \overline{5} \pmod{17}$$

If a and b are integers in the range [0, N − 1], then their sum is in the range [0, 2N − 2] and their difference is in the range [−N + 1, N − 1], so determining the representative in [0, N − 1]requires at most one subtraction or addition (respectively) of N. However, the product ab is in the range [0, N^2 − 2N + 1]. Storing the intermediate integer product ab requires twice as many bits as either a or b, and efficiently determining the representative in [0, N − 1] requires division. Mathematically, the integer between 0 and N − 1 that is congruent to ab can be expressed by applying the division algorithm:

$$ab = qN + r,$$

where q is the quotient $\lfloor ab/N \rfloor$ and r, the remainder, is in the interval [0, N − 1]. The remainder r is ab mod N. Determining r can be done by computing q, then subtracting qN from ab. For example, the product $\overline{7} \cdot \overline{15}$ is determined by computing

$$7 \cdot 15 = 105$$

dividing $\lfloor 105/17 \rfloor = 6$, and subtracting $105 - 6 \cdot 17 = 105 - 102 = 3$.

The only mathematical requirement on the auxiliary modulus R is that it be a positive integer such that gcd(N, R) = 1. For computational purposes it is also necessary that division and reduction modulo R be inexpensive, and the modulus is not useful for modular multiplication unless R > N. The Montgomery form or Montgomery representation of the residue class $\bar{a}$with respect to R is aR mod N, that is, it is the representative of the residue class $\overline{aR}$. For example, suppose that N = 17 and that R = 100. The Montgomery forms of 3, 5, 7, and 15 are300

mod 17 = 11, 500 mod 17 = 7, 700 mod 17 = 3, and 1500 mod 17 = 4. Addition and subtraction in Montgomery form are the same as ordinary modular addition and subtraction because of the distributive law:

$$aR + bR = (a + b)R,$$
$$aR - bR = (a - b)R.$$

This is a consequence of the fact that, because gcd(R, N) = 1, multiplication by R is an isomorphism on the additive group Z/NZ. For example, (7 + 15) mod 17 = 5, which in Montgomery form becomes (3 + 4) mod 17 = 7. Multiplication in Montgomery form, however, is seemingly more complicated. The usual product of aR and bR does not represent the product of a and b because it has an extra factor of R:

$$(aR \bmod N)(bR \bmod N) \bmod N = (abR) R \bmod N.$$

Computing products in Montgomery form requires removing the extra factor of R [4]. While division by R is cheap, the intermediate product is not divisible by R because the modulo operation has destroyed that property. So for instance, the product of the Montgomery forms of 7 and 15 modulo 17 is the product of 3 and 4, which are 12. Since 12 is not divisible by 100, additional effort is required to remove the extra factor of R. Removing the extra factor of R can be done by multiplying by an integer R′ such that

$$RR' \equiv 1 \pmod N,$$

that is, by an R′ whose residue class is the modular inverse of R mod N. Then, working modulo N,

$$(aR \bmod N)(bR \bmod N)R' \equiv (aR)(bR)R^{-1} \equiv (ab)R \pmod N.$$

The integer R′ exists because of the assumption that R and N are co-prime. It can be constructed using the extended Euclidean algorithm. The extended Euclidean algorithm efficiently determines integers R′ and N′ that satisfy Bézout's identity 0 < R′ < N, 0 < N′ < R, and:

$$RR' - NN' = 1.$$

This shows that it is possible to do multiplication in Montgomery form. A straightforward algorithm to multiply numbers in Montgomery form is therefore to multiply aR mod N, bR mod N, and R′ as integers and reduce modulo N.

V. ARCHITECTURES OF SYSTEM

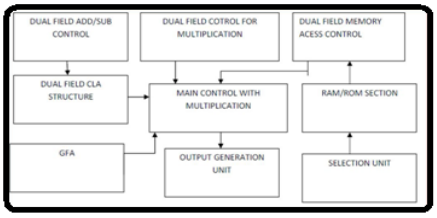

Fig.1.Proposed Architecture

Dual-Field Addition /Subtraction: A Dual-field Full Adder (DFA) cell is basically a Full Adder (FA) cell consists of half adder and equipped with a field select signal (fsel), that controls the operation mod. When fsel= 0, the carry output is forced to 0 and the sum outputs the XOR operation of the inputs. As already mentioned, this is equivalent to the addition operation in GF(2n). When fsel = 1, GF(p) mode is selected and the cell operates as a normal FA cell. Dual-field adders in various configurations can be mechanized by utilizing DFA cells. In the proposed implementation, 3-level, CLA with 4-bit Carry Look ahead Generator (CLG) groups are employed. An example of a 4-bit dual-field CLA is shown below. The GAP modules generate the signals pi = xi XOR yi , gi = xi AND yi , ai = xi OR yi , and AND gates along with a fsel signal control whether to eliminate carries or not. The carry look ahead generator is an AND -OR network. Comparing the previous approach employing, which requires L (L-1)/2 modular multiplications, the optimized MRC requires only L-2 modular multiplications [5]. The methodology is further extended for the case of GF (2n). With trivial modifications of algorithms for modular addition/subtraction in a dual-field modular adder/subtracted (DMAS) can be mechanized using CLA adders When , the circuit is in mode and the output is derived directly from the top adder which performs a addition.

Dual-Field Multiplication: A parallel multiplier, which is suitable for high-speed arithmetic, and requires little modification to support both fields, is considered in the proposed architecture. Regarding input operands, either integers or polynomials, partial product generation is common for both fields. Consequently, the addition tree that sums the partial products must support both formats. In GF(2^n) mode, if DFA cells are used, all carries are eliminated and only xor operations are performed among partial products. In GF(p) mode, the multiplier acts as a conventional tree multiplier. A 4 X 4-bit example of the proposed dual-field multiplier (DM) with output in carry-save format is depicted.

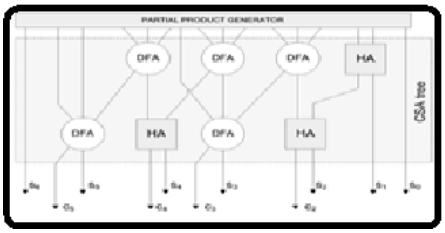

Fig.2.Dual-field multiplier (DM)

Dual-Field Modular Reduction: A final modular reduction by each RNS/PRNS modulus is required, for each multiplication outcome, within each MAC unit. Dual-field modular reduction unit (DMR) be simplified as

$$\langle c \rangle_{p_i} = \begin{cases} \sum_{i=0}^{r-1} \xi_i 2^i, \xi < 2^r - \mu \\ \sum_{i=0}^{r} \xi_i 2^i + \mu_i, 2^r - \mu < \xi < 2^r \end{cases}$$

The same decomposition can be applied to polynomials and consequently, if dual-field adders and dual-field multipliers are employed. The word length of can be limited to a maximum of 10 bits for a base with 66 elements.

MAC Unit: The circuit organization from the suggested MAC unit operation is examined below in three steps, akin to the 3 phases from the calculations it handles, RNS/PRNS Montgomery multiplication, and residue-to-binary conversion. i) Binary-to-Residue Conversion: Initially, r-bit words from the input operands, as implied by cascaded to every MAC unit and kept in RAM1. These words function as the very first input towards the multiplier, combined with the quantities that are kept in a ROM. Their multiplication creates the inner products) that are added recursively within the DMAS unit. It makes sense stored through the bus in RAM1. The operation is repeated for that second operand and it makes sense kept in RAM2, to ensure that once the conversion is completed, each MAC unit supports the residue digits of these two operands within the two RAMs. The conversion requires steps to become performed. ii) Montgomery Multiplication: The initial step from the suggested DRAMM is really a modular multiplication from the residue digits from the operands. As these digits are immediately available through the two RAMs, a modular multiplication is performed and also the lead to is kept in RAM1 for base and RAM2 for base . Step Two of DRAMM is really a multiplication from the previous result having a constant supplied by the ROM. The outcomes match inputs from the DBC formula and therefore are stored again in RAM1. All MAC units are updated with the bus using the corresponding RNS digits of other MACs along with a DBC process is initiated.

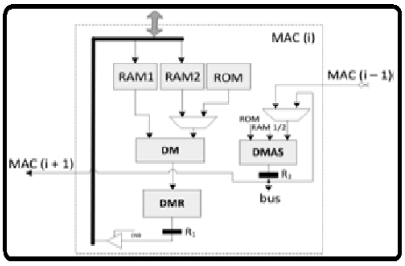

Fig.3.Proposed MAC Unit

iii) Residue-to-Binary Conversion: Residue-to-binary conversion is basically repeating the DBC formula, which aren't modulo operations. As one example of alteration, assume the generation from the inner products in row 1 Each method is calculated in parallel in every MAC unit along with a "carry-propagation" from MAC(1) to is conducted to include all inner products [6]. When summation finishes the very first digit of it makes sense created in parallel with this particular "carry-propagation", the interior products of line 2 are calculated. When a MAC unit completes an inclusion of carry-propagated inner products for line 1, a brand new addition for line 2 is conducted.

VI. CONCLUSION

The mathematical framework along with a flexible, dual-field, residue arithmetic architecture for Montgomery multiplication is developed and also the necessary conditions for that system parameters are derived. The suggested DRAMM architecture supports all operations of Montgomery multiplication residue-to-binary and binary-to-residue conversions, MRC for integers and polynomials, dual-field modular exponentiation and inversion, within the same hardware. Generic complexity and real performance comparisons with condition-of-the-artworks prove the potential for residue arithmetic exploitation in Montgomery multiplication.

VII. REFERENCES

[1] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curves Cryptography. New York, NY, USA: Springer-Verlag & Hall/CRC, 2004.

[2] J.-P. Deschamps, Hardware Implementation of Finite-Field Arithmetic. New York, NY, USA: McGraw-Hill, 2009.

[3] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems,"Commun. ACM,vol.21,pp. 120–126, Feb. 1978.

[4] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-Rower architecture for fast parallel Montgomery multiplication," in EUROCRYPT'00: Proc. 19th Int. Conf. Theory and Application of Cryptographic Techniques, 2000, pp. 523–538.

[5] D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas, and T. Stouraitis, "An RNS implementation of an elliptic curve point multiplier,"IEEE Trans. Circuits Syst. I, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[6] I.Blake,G.Seroussi,andN.Smart, Elliptic Curves in Cryptography. Cambridge, U.K.: Cambridge Univ. Press, 2002.

