www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 9 Sep 2015, Page No. 14307-14312

Analytical Approach For Enhanced Residue Modular Multiplier For Cryptography

Afshan Fathima (Pg Scholar) [1] *B.Sanjai Prasada Rao Associate Professor*[2]

Department of ECE, Lords Institute of Engineering and Technology, Hyderabad, INDIA

Abstract

This paper presents an implementation of VLSI architecture for Dual Field Residue Arithmetic modular multiplier with less delay based on finite field arithmetic to support all public key cryptographic applications. A new method for incorporating Residue Number System (RNS) and Polynomial Residue Number system (PRNS) in modular multiplication is derived and then implemented VLSI Architecture for dual field residue arithmetic modular multiplier with less delay. This architecture supports the conversions, modular multiplication for polynomials and integers and modular exponentiation in same hardware. This architecture has a carry save adders (CSAs) and parallel prefix adders in MAC units to speed up the large integer arithmetic operations over GF (P) and GF (2n), hence this reduces the delay up to 10 percent.

KEYWORDS: Finite field arithmetic, Residue number and Polynomial Residue number systems, modular arithmetic, parallel arithmetic and logic structures, and Montgomery multiplication

1. INTRODUCTION

Now a days, many of applications including cryptography, error correction coding, computer algebra, DSP, etc., depends on the efficient realization of arithmetic over finite fields of the form GF (2n), where n € Z and n ≥ 1, or the form GF(P) , where P is a prime. Special case of multiplications are formed by Cryptographic applications, since, for security reasons, they require large integer operands. Almost all public key cryptography, such as Elliptic Curves Cryptography (ECC) and RSA cryptography employ modular multiplication with very large numbers, so faster modular multiplication has become an important cryptography issue. For achieving satisfactory cryptosystem performance, efficient field multiplication with large operands is crucial since multiplication is the most time and area consuming operation. Therefore, there is a need for increasing the speed of cryptosystems employing modular arithmetic with the least possible area penalty.

The perfect approach to achieve this would be through parallelization of their operations. The RNS/PRNS is a non-weighted number system which speeds up arithmetic operations by dividing them into smaller parallel operations, and they provide interesting low power architecture. Since the RNS/PRNS system is not a positional number system where each digit corresponds to a certain weight, it is hard to implement the operations of comparison and division. RNS/PRNS is one of the most popular techniques for reducing the power dissipation and the computation load in VLSI systems design. On the other hand, for RNS/PRNS implementations, the extra cost of input converters to translate numbers from a standard binary format into residues and output converters to translate from RNS/PRNS to binary representations are needed.

A new methodology for embedding residue arithmetic in a dual field Montgomery modular multiplication algorithm for integers in and for polynomials in is presented in this paper. The derived architecture is highly parallelizable

DOI: 10.18535/ijecs/v4i9.42

and versatile, as 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 in the same hardware.

2. CONVENTIONAL WORKS

GF (2n) implementation has been progressed a lot in these days. PRNS incorporation in field multiplication based on a straightforward implementation of the Chinese Remainder Theorem (CRT) for polynomials is implemented, requires large storage resources and many pre-computations. The multipliers perform multiplication in PRNS are proposed, but the result is then converted back to polynomial representation.

This limitation makes these methods inappropriate for cryptographic algorithms, since it requires consecutive multiplications. Finally, algorithm which employs trinomials for the modulus set and performs PRNS Montgomery multiplication has been proposed. But has no reference to conversion methods and the trinomials requirement may issue limitations in the PRNS data range.

GF (P) implementations have also withstood great analysis, with the Montgomery algorithm being used in the majority of them. Montgomery multiplication designs fall into two categories. The first includes fixed-precision input operand implementations, in which the multiplicand and modulus are processed in full world length, while multiplier is handled bit-by-bit. These designs are optimized for certain word lengths and do not scale efficiently for departures from these word lengths. Their performance has been improved by high-radix algorithms and architectures.

The second category includes scalable architectures for variable word-length operands, based on algorithms, in which the multiplicand and modulus are processed word by word; while the multiplier is consumed bit by bit. Montgomery's algorithm has also become a predicate for dual-field implementations. The Montgomery architectures perform well for RSA key word lengths, by processing word size data, since RSA key sizes (512, 1024, 2048, etc.) are always multiples of word size. However, in ECC, key sizes are not integer multiples of word size, meaning that, if this architecture were to be used in ECC, architecture configured at bit-level overcomes this problem.

3. RESIDUE NUMBER SYSTEM

In recent years, we have experienced a great development in the field of digital communication technologies which brought together a great concern about security in computers and communications systems. Several public-key cryptosystems were proposed in order to enable the encryption of messages using a public encryption key 'e' without a prior communication of a secret key. The secrecy relies on the fact that decryption key is computationally infeasible to deduce from the public encryption key. Then, the only person who can decrypt the cipher-text is the receiver, who knows the secret decryption key.

Public-key cryptography plays an important role in digital communication and storage systems. Processing public-key cryptosystems requires huge amount of computation, and, there is therefore, a great demand for developing dedicated hardware to speed up the computations. Speeding up the computation using specialized hardware enables the use of larger keys in public-key cryptosystems. This is translated into an increase of the security of the system. Also, this enables the speedup of a secure link between two distant points using an insecure channel, which is critical in real-time systems. The reduction of the hardware amount is another important aspect when implementing in dedicated hardware because it allows for the miniaturization of portable devices and reduces fabrication costs.

The Residue Number System (RNS) is a non-weighted number system that can map large numbers to smaller residues, without any need for carry propagations .Its most important property is that additions, subtractions, and multiplications are inherently carry-free. These arithmetic operations can be performed on residue digits concurrently and independently. Thus, using residue arithmetic, would in principle, increase the speed of computations RNS has shown high efficiency in realizing special purpose applications such as digital filters , image processing , RSA cryptography and specific applications for which only additions, subtractions and multiplications are used and the number dynamic range is specific. Special moduli sets have been used extensively to reduce the hardware complexity in the implementation of converters and arithmetic operations. Among which the triple moduli set {2n+1, 2n, 2n-1} have some benefits. Since the operation of multiplication is of major importance for almost all kinds of processors, efficient implementation of multiplication modulo 2n-1 is important for the application of RNS.

4. PROPOSED METHOD

4.1 Residue number system

RNS consists of a pair wise relatively prime integers set $\mathcal{A} = (m_{1,}m_2, \dots, m_L)$ and simultaneously RNS range is computed as $A = \prod_{i=1}^{L} m_i$. A unique RNS representation $Z_{\mathcal{A}}$which denotes the two integers a, b in RMS format i.e., $a_A = (a_1, a_{2,\dots,}a_L)$ and $b_A = (b_1, b_{2,\dots,}b_L)$given by $Z_{\mathcal{A}}$=$(z_1, z_2, \dots, z_L)$=$\left((z)m_{1,}(z)m_{2,\dots,}(z)m_{L,}\right)$, then at the end, one can accomplished the process in parallel was as shown below

$a_{\mathcal{A}} \otimes b_{\mathcal{A}}$ = $\left(\langle a_1 \otimes b_1\rangle_{m_{1,}}\langle a_2 \otimes b_2\rangle_{m_{1,\dots,}}\langle a_L \otimes b_L\rangle_{m_{L,}}\right)$ (1)

To reconstruct the integer from its residues, two methods may be employed. The first is through the CRT according to

$z = \sum_{i=1}^{L}\langle z_i . A_i^{-1}\rangle_{m_i} . A_{i-}\gamma A$ (2)

Where $A_i - {}^{A}/_{m_{,i}}$, A_i^{-1} is the inverse of A_i modulo m_i , and γ is an integer correction factor

The second method is through the MRC. The MRC of an integer with an RNS representation $Z_{\mathcal{A}}$=$(z_1, z_2, \dots, z_L)$ is given by

$z = U_1 + W_2 u_2 + \cdots + W_L U_L$ (3)

Where $W_i - \prod_{j=1}^{i-1} m_j, \forall i \epsilon [2, L]$ and $U_i s$ are4 according to the

$$U_1 = Z_1$$

$$U_2 = \langle\langle Z_2 - Z_1\rangle\rangle m_2$$

$$U_3 = \langle z_3 - z_1 - W_2 U_2\rangle m_3$$

$$\vdots$$

$$U_L = \langle z_L - z_1 - W_2 U_2 - W_3 U_3 - \dots - W_{L-1} U_{L-1}\rangle m_L \quad (4)$$

4.2 Polynomial Residue Number System

Similar to RNS, a Polynomial Residue Number System (PRNS) is defined through a set of pair wise relatively prime polynomials$A = m_1 x, m_2 x, \dots\dots\dots\dots . . m_L x$. We denote by $A(x) = \prod_{i=1}^{L} m_i\,(x)$ the dynamic range of the PRNS. In PRNS, every polynomial has a unique PRNS representation: $Z_{\mathcal{A}}$=$(z_1, z_2, \dots, z_L)$ such as $z_i = z(i) mod$mod mi x, i ∈ [1, L], denoted as $< z > m_i$.In the rest of the paper, the notation "(x)" to denote polynomials shall be omitted, for simplicity. The notation z will be used interchangeably to denote either an integer or a polynomial, according to context according to the context. In the PRNS representation all operations can be performed in parallel. Conversion from PRNS to weighted polynomial representation is identical to the MRC for integers. The only difference is that, the subtractions in () are substituted by polynomial additions.

4.3 Montgomery Multiplication

A.GF (P) arithmetic

Field elements GF (P) in are integers in [0 to P-1] and arithmetic is performed modulo P. Since Montgomery's method was originally devised to avoid divisions, it is well suited to RNS implementations, considering that RNS division are inefficient to perform**.**

Algorithm 1 Montgomery Modular Multiplication

```
Input: a, b, p, R,$R^{-1}/* a, b <$ p
Output: c≡ $abR^{-1}$  mod p, l* c < 2p
1: s ← $a \cdot b$
2: t← $s \cdot (-p^{-1})$  mod R
3: u ← $t \cdot p$
4: v ← $s + u$
```

5: c $\leftarrow v/R$

B. GF (2^n) arithmetic

In GF (2n) arithmetic, field elements are polynomials are represented as vectors with dimension n ,relative to a given polynomial basis 1, α, α^2 , … … . . , $\alpha^{n-1,}$ where α is a root of an irreducible polynomial p of degree n over GF(2). The addition of two polynomials a and b in GF (2 n) is performed by adding the their coefficients i.e., modulo 2.The multiplication of two polynomials is

c = a. b mod p

Algorithm 2 RNS Montgomery Multiplication (RMM)

```
Input: $a_T$, $b_T$, $(-p^{-1})_B$, $Q_A^{-1}$, $p_{A,}$/* $a, b < 2p$
Output: $c_T$ , l* c < 2p and $c \equiv abQ^{-1}$ mod p
1: $s_T \leftarrow a_T \cdot b_T$
2: $t_B \leftarrow s_B \cdot (-p^{-1})_B$
3: $t_A \leftarrow t_B$/* base conversion step
4: $u_A \leftarrow t_A . p_A$
5: $v_A \leftarrow s_A + u_A$
6: $c_A \leftarrow v_A . Q_A^{-1}$
7: $c_B \leftarrow c_A$/* base conversion step
```

5. CONVERSIONS

Let us consider A= $(p_1, p_2, p_3 \ldots \ldots \ldots \ldots \ldots, p_L)$ as base, this shall be used to analyze the Conversions to/from residue representations.

1. Binary-to-Residue Conversion

A radix- representation of an integer z as an L- tuple

$(Z^{(L-1)} \ldots (Z^{(0)})$ satisfies

z = $\sum_{i=0}^{L-1} z^{(i)} 2^{ri}$ (5)

Where, $0 \le z^{(i)} \le 2^r - 1$

To compute z_Aa method is devised, the multiply and Accumulate structure in DRAMM is implemented for this method. By applying the modulo pj operation, we obtain

$<z>_{P_j} = <\sum_{i=0}^{L-1} z^{(i)} < 2^{ri} >_{P_j}>_{P_j}, \forall j \in [1, L]$ (6)

If constants $<2^{ri}>_{P_j}$are precomputed, this computation is well suited to the proposed MAC structure and can be computed in L steps, when executed by units in parallel. Similar to the integer case, a polynomial lz(x) ∈ GF (2^n) can be written as

z = $\sum_{i=0}^{L-1} z^{(i)} x^{ri}$ (7)

Applying the modulo P_j operation in the above equation

$<z>_{P_j} = <\sum_{i=0}^{L-1} z^{(i)} < x^{ri} >_{P_j}>_{P_j}, \forall j \in [1, L]$ (8)

Which is a similar operation to operation for integers, if $x^{ri} >_{P_j}$are pre-computed. From the above analysis conversions in both fields can be unified into a common conversion method, if dual-field circuitry is employed.

2. Residue-to-Binary Conversion

As all operands in (4) are of word length, they can be efficiently handled by an rbit MAC unit. However, employs multiplications with large values, namely the W_I s. To overcome this problem can be rewritten as matrix notation. The inner products of a row are calculated in parallel in each MAC unit. Each MAC then propagates its result to subsequent MACs, so that at the end the last MAC (L) outputs the radix- 2 rdigit z (i) of the result. In parallel with this summation, inner products of the next row i+1 can be formulated, since the adder and multiplier of the proposed MAC architecture may operate in parallel.

6. SIMULATION RESULTS

Figure 1: **SIMULATION RESULT OF MULTI RESIDUE TEST**

Figure 2: **SIMULATION RESULT OF DUAL CLA TEST**

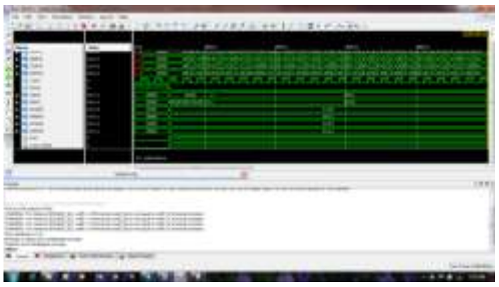

Figure 3: **SIMULATION RESULT OF DUAL FIELD MULTI TEST**

Figure 4: **SIMULATION RESULT OF FOUR MAC TEST**

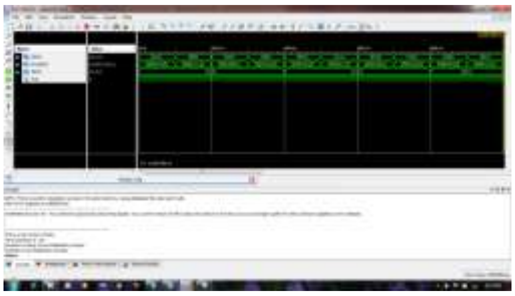

Figure 5: **SIMULATION RESULT OF MODULAR REDUCTION TEST**

Figure 6: **SIMULATION RESULT OF MUX LOGIC TEST**

7. CONCLUSION

An Efficient high speed RNS modular multiplier implemented in this paper ,that operates in both GF p and GF(2 n) arithmetic fields and necessary conditions for the system parameters are mentioned. The DRAMM architecture supports all operations of Montgomery multiplication, residue-to-binary conversion and binary-to-residue conversion, MRC for polynomials and integers and modular exponentiation in same hard ware. The MAC units in DRAMM architecture reduces the delay, hence this is suited for high speed applications like all types of public key cryptography and DSP etc.

REFERENCES

[1] N. Szabo and R. Tanaka, Residue arithmetic and its applications to computer technology. McGraw-Hill, 1967.

[2] R. Szerwinski and T. G ¨ uneysu, “Exploiting the power of GPUs for asymmetric cryptography,” Cryptographic Hardware and Embedded Systems–CHES 2008, pp. 79–99, 2008.

[3] P. Montgomery, “Modular multiplication without trial division,” Mathematics of computation, vol. 44, no. 170, pp. 519– 521, 1985.

[4] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, “Cox-rower architecture for fast parallel Montgomery multiplication,” in Advances in Cryptology EUROCRYPT 2000, ser. LNCS. Springer Berlin / Heidelberg, 2000, pp. 523–538.

[5] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, “Implementation of RSA algorithm based on RNS Montgomery multiplication,” in Cryptographic Hardware

and Embedded Systems CHES 2001, ser. LNCS. Springer Berlin / Heidelberg, 2001, pp. 364–376.

[6] J.-C. Bajard, L.-S. Didier, and P. Kornerup, “An RNS Montgomery modular multiplication algorithm,” Computers, IEEE Transactions on, vol. 47, no. 7, pp. 766 –776, jul. 1998.

[7] J.-C. Bajard, L. S. Didier, and P. Kornerup, “Modular multiplication and base extensions in residue number systems,” in Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on, 2001, pp. 59–65.

[8] A. Shenoy and R. Kumaresan, “Fast base extension using a redundant modulus in RNS,” Computers, IEEE Transactions on, vol. 38, no. 2, pp. 292–297, Feb 1989.

[9] J.-C. Bajard and L. Imbert, “A full RNS implementation of RSA,” Computers, IEEE Transactions on, vol. 53, no. 6, pp. 769–774, June 2004.

[10] N. Guillermin, “A high speed coprocessor for elliptic curve scalar multiplications over Fp,” in Workshop on Cryptographic Hardware and Embedded Systems 2010 (CHES 2010), ser. LNCS. Springer Berlin / Heidelberg, 2010, pp. 48–64.

[11] F. Gandino, F. Lamberti, J.-C. Bajard, and P. Montuschi, “Preprocessing in RNS Montgomery multiplication,” Tech. Rep., 2010.

[12] M. Pohst and H. Zassenhaus, Eds., Algorithmic algebraic number theory. New York, NY, USA: Cambridge UniversityPress, 1989, ch. 2.2.5.

[13] J. Bajard, N. Meloni, and T. Plantard, “Efficient RNS bases for cryptography,” in IMACS’05 : World Congress: Scientific Computation, Applied Mathematics and Simulation, July 2005.

[14] D. Gajski, Principles of Digital Design. Prentice-Hall, 1997.