ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

Multifunction Residue Architectures for Cryptography

C.Gayathri, R.sanjay Babu

M.Tech Student (VLSI), Department of ECE, Shree institute of Technology, Tirupathi, India

Assistant Professor, Department of ECE, Shree institute of Technology, Tirupathi, India

ABSTRACT: The proposed procedure for fusing Residue Number System (RNS) and Polynomial Residue Number System (PRNS) in Montgomery particular increase in GF (p) or GF (2n) individually, and VLSI structural engineering of a double field deposit number-crunching Montgomery multiplier are introduced in this paper. An investigation of info/yield changes to/from deposit representation, alongside the proposed buildup Montgomery duplication calculation, uncovers normal increase gather information ways both between the converters and between the two buildup representations.

An adaptable structural engineering is determined that backings all operations of Montgomery augmentation in GF (p) and GF(2n), info/yield changes, Mixed Radix Conversion (MRC) for whole numbers and polynomials, double field secluded exponentiation and reversal in the same equipment. Point by point correlations with best in class executions demonstrates the capability of buildup math abuse in double field particular increase.

KEYWORDS: Computations in finite fields, computer arithmetic, Montgomery multiplication, parallel arithmetic and logic structures.

I. INTRODUCTION

A noteworthy number of utilizations including cryptography, blunder amendment coding, PC variable based math, DSP, and so forth., depend on the productive acknowledgment of number-crunching over limited fields of the structure GF(2n) , where n ε Z and n > 1, or the structure GF(P) , where P a prime. Cryptographic applications frame an exceptional case, subsequent to, for security reasons, they oblige huge number operands [1]–[5]. Effective field augmentation with substantial operands is essential for accomplishing a delightful cryptosystem execution, since increase is the most time-and territory devouring operation.

In this manner, there is a requirement for expanding the pace of cryptosystems utilizing measured number juggling with the slightest conceivable range punishment. An undeniable way to deal with accomplish this would be through parallelization of their operations. Lately, RNS and PRNS have appreciated replenished investigative enthusiasm because of their capacity to perform quick and parallel secluded number-crunching [6]–[13]. Utilizing RNS/PRNS, a given way serving a vast information extent is supplanted by parallel ways of littler element ranges, with no requirement for trading data between ways. Thus, the utilization of buildup frameworks can offer decreased many-sided quality and force utilization of number-crunching units with vast word lengths [14].On the other hand, RNS/PRNS usage bear the additional expense of data converters to interpret numbers from a standard twofold configuration into deposits and yield converters to decipher from RNS/PRNS to parallel representations [14].

Another approach for implanting buildup math in a double field Montgomery measured increase calculation for whole numbers in and for polynomials in is displayed in this paper. The numerical conditions that should be fulfilled for a substantial RNS/PRNS consolidation are inspected. The inferred construction modeling is exceptionally parallelizable and adaptable, as it backings double to-RNS/PRNS and RNS/PRNS-to-paired transformations, Mixed Radix Conversion (MRC) for whole numbers and polynomials, double field Montgomery duplication, and double field secluded exponentiation and reversal in the same equipment. Whatever remains of the paper is sorted out as takes after. A brief diagram of related past work is offered in Section II, while the fundamental ideas of RNS and PRNS are condensed in Section III. In Section IV, fundamental limited field number juggling ideas are given and the operation of field augmentation is characterized. Taking after and Montgomery duplication calculations are introduced. In Section

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

VI, the proposed RNS/PRNS Montgomery increase calculation is broke down. The numerical conditions that permit a substantial joining of buildup number-crunching in the Montgomery calculation are additionally exhibited.

We briefly discuss related works in Section I, while Section II presents an overview of the Existing schemes. The proposed system along with possible hardware implementations and their analysis are described in Section III. In Section IV, the results .Finally, this paper is concluded in Section VI.

II. EXISTING SYSTEM

Essential advancement has been accounted for recently with respect to GF(2n) usage. The Massey-Omura calculation [15], the presentation of ideal ordinary bases [16] and their product and equipment usage [17], [18], the Montgomery calculation for duplication in GF(2n) [19], and in addition PRNS application in GF(2n) augmentation, are, among others, critical advances [9], [10], [12], [20], [21]. PRNS fuse in field duplication, as proposed in [9], is in view of a direct usage of the Chinese Remainder Theorem (CRT) for polynomials which obliges huge capacity assets and numerous pre-reckonings. The multipliers proposed in [10], [20], perform increase in PRNS, yet the outcome is changed over back to polynomial representation.

This constraint makes them improper for cryptographic calculations which require continuous duplications. At long last, a calculation which utilizes trinomials for the modulus set and performs PRNS Montgomery duplication has been proposed [12]. Be that as it may, there is no reference to transformation techniques and the utilization of trinomials may issue restrictions in the PRNS information range. GF(p) executions have likewise withstood awesome investigation, with the Montgomery calculation being utilized as a part of the greater part of them. Montgomery increase plans fall into two classifications. The principal incorporates settled exactness information operand executions, in which the multiplicand and modulus are prepared in full word length, while the multiplier is taken care of a tiny bit at a time. A structural engineering arranged at bit-level conquers this issue [23]. At long last, techniques for installing RNS in Montgomery increase have additionally been proposed.

A. Residue Arithmetic

i. Residue Number System

RNS consists of a set of, pair-wise relatively prime integers $A=(m_1,m_2,\ldots\ldots m_L)$ (called the *base*) and the range of the RNS is computed as $A=\prod_{i=1}^{L} m_i$. Any integer has a unique RNS representation Z_A given by $Z_A=(Z_1,Z_2,\ldots.Z_L)=\{(Z)m_1,(Z)m_2,\ldots\ldots(Z)m_L\}$, where$(Z)m_i$ denotes the operation Z mod m_i . Assuming two integers a ,b in RNS format, i.e. $a_A=(a_1,a_2,\ldots a_L)$ and $b_A=(b_1,b_2,\ldots b_L)$, then one can perform the operations $\otimes \in (+,-,*)$ in parallel by

$$a_A \otimes b_A = \left(\langle a_1 \otimes b_1\rangle_{m_1}, \langle a_2 \otimes b_2\rangle_{m_2}, \ldots, \langle a_L \otimes b_L\rangle_{m_L}\right) .1$$

To reconstruct the integer from its residues, two methods may be employed [14]. The first is through the CRT according to

$$z = \sum_{i=1}^{L} \langle z_i \cdot A_i^{-1}\rangle_{m_i} \cdot A_i - \gamma A$$

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 z with an RNS representation $Z_A=(Z_1,Z_2,\ldots.Z_L)$ is given by

$Z=U_1+W_2U_2+\ldots\ldots W_LU_L$

Where $W_i = \prod_{j=1}^{i-1} m_j, \forall i \in [2,L]$ and the U_i S are computed according to

$U_1= z_1$

$U_2= (Z_2 - Z_1)m_2$

$U_3= (Z_3 - Z_1-W_2U_2)m_3$

.

$U_L= (Z_L - Z_1-W_2U_2- W_3U_3-\ldots\ldots- W_{L-1}U_{L-1})m_L$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

ii. Polynomial Residue Number System

Similar to RNS, a PRNS is defined through a set of L, pair-wise relatively prime polynomials $\mathcal{A} = (m_1(x), m_2(x), \ldots, m_L(x))$. We denote by $A(x) = \prod_{i=1}^{L} m_i(x)$ the dynamic range of the PRNS. In PRNS, every polynomial $z(x) \in GF(2^n)$, , with, has a unique PRNS representation:

$z_{\mathcal{A}} = (z_1, z_2, \ldots, z_L)$

Such as $z_i = z(x) \bmod m_i(x), i \in |1, L|$. denoted as Z (mi)

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 Z or a polynomial Z(x).

III. PROPOSED SYSTEM

A. Montgomery Multiplication

i. GF(p) Arithmetic

Field elements in GF(p) are integers in [0,p] arithmetic is performed modulo. Montgomery's algorithm for modular multiplication without division [43] is presented below, as Algorithm 1, in five steps, where R is the Montgomery radix, $\gcd(R, p) = 1, \text{and } p < R.$, and . R must be chosen so

Algorithm 1 Montgomery Modular Multiplication

Input: a, b, p, R, R^{-1} /* $a, b < p$
Output: $c \equiv abR^{-1} \mod p$, /* $c < 2p$
1: $s \leftarrow a \cdot b$
2: $t \leftarrow s \cdot \left(-p^{-1}\right) \mod R$
3: $u \leftarrow t \cdot p$
4: $v \leftarrow s + u$
5: $c \leftarrow v/R$

that steps 2 and 5 are efficiently computed. It is usually chosen to be a power of 2, when radix-2 representation is employed. 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.

ii. GF (2^n)Arithmetic

Field elements in GF(2^n)are polynomials represented as binary vectors of dimension n , relative to a given polynomial basis $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$, where is a α root of an irreducible polynomial p of degree n over GF(2) . The field is then realized as $GF(2)[x]/(p)$ and the arithmetic is that of polynomials of degree at most n-1, modulo p [1].

The addition of two polynomials a and b in GF(2^n) is performed by adding the polynomials, with their coefficients added in GF(2) , i.e., modulo 2. This is equivalent to a bit-wise XOR operation on the vectors a and b. The product of two elements a and b in GF(2^n) is obtained by computing

C = a.b mod p

Where c is a polynomial of degree at most n-1 and $c \in GF(2^n)$. A Montgomery multiplication algorithm suitable for polynomials in GF(2^n) has been proposed [19]. Instead of computing the product c= a.b mod p ,

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

the algorithm computes C= $a \cdot b \cdot R^{-1} \bmod p$, with $\deg\{c(x)\} < n$, with and R is a special fixed element in GF(2^n) . The selection of $R(x) = x^n$ is the most appropriate, since modular reduction and division by x^n are simple shifts [3]. The algorithm is identical to Algorithm 1, except from the constant $-p^{-1}$ in step 2, which is in GF(2^n) .

B. New Methodology for Embedding Residue Arithmetic in Montgomery Multiplication

i. Embedding RNS in GF(P) Montgomery Multiplication

An MRC-based algorithm [44] that avoids the evaluation of the factor of (2) forms the basis of the proposed RNS-based Montgomery multiplication algorithm. The derived algorithm is briefly presented here as Algorithm 2, and extended for the case of GF(2^n). Two RNS bases are introduced, namely $\mathcal{A} = (p_1, p_2, \ldots, p_L)$ and $\mathcal{B} = (q_1, q_2, \ldots, q_L)$, such that $\gcd(p_i, q_j) = 1, \forall i, j \in$ .The 5 steps of the Montgomery algorithm are translated to RNS computations in both bases, denoted from now on as $\mathcal{T} = \mathcal{A} \bigcup \mathcal{B}$.

Nevertheless, the computations in base cannot be continued for steps 3, 4, and 5 of Algorithm 1, since in step 5 we would need to compute a quantity of the form , which does not exist since s are factors of . Thus, a base conversion (BC) step, from base to base , is embedded, to register . is then used to execute the old steps 3, 4, and 5 in base . The outcome toward the end of this calculation is an amount in RNS design that equivalents , since BC is sans slip. In Algorithm 2, inputs and yield are all not exactly, so they are good with one another. This permits the development of a secluded exponentiation calculation by redundancy of the RNS Montgomery increase. Base transformation in step 7 is used for the same reason. Calculation 3 portrays the proposed base change handle that changes over a whole number communicated in RNS base as to the RNS representation of another base.

Rather than different RNS Montgomery augmentation calculations which additionally utilize MRC [39], [40], [44], the proposed one executes a rearranged variant of the MRC in (3) and (4) which, as will be demonstrated in next areas, not just lessens the aggregate many-sided quality of the calculation additionally offers better open doors for parallelization of operations.

ii. Embedding PRNS in GF(2^n) Montgomery Multiplication

A change of the Montgomery calculation for augmentation in GF(2n) that incorporates PRNS is proposed next. The proposed calculation utilizes general polynomials of any degree, and is an expansion of a calculation [12], which utilizes trinomials for the PRNS modulus set. Moreover, the proposed calculation addresses the issue of changing over information to/from PRNS representation. As opposed to a comparable calculation in [45], which utilized CRT for polynomials for the BC calculation, the proposed construction modeling utilizes MRC. This takes into consideration double field RNS/PRNS execution, which is not bolstered in [45], and another system to actualize RNS-to-parallel transformation as will be demonstrated in Section V-D.

The proposed calculation for PRNS Montgomery augmentation (PRMM) is exhibited underneath as Algorithm 4. The relating calculation for base change in is indistinguishable to Algorithm 3, in this manner it is overlooked for straightforwardness reasons. The main contrast is that number augmentations/subtractions and duplications are supplanted by polynomial ones. Once more, the level of data and yield polynomials are both not exactly, which permits the development of a particular exponentiation calculation by redundancy of the PRMM. Base change in step 7 is utilized for the same reason.

iii. Proof of PRMM Algorithm's Validity

Theorem 1:

If gcd{P,Q}=1 , gcd{Q,P}=1(2),deg(P)<n (3) , and deg(Q)>n (4) , then Algorithm 4 outputs C_T , for which $c=abQ^{-1}$ mod P and deg{c}<n.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

Proof:

Since gcd{P,Q}=1 and gcd{Q,P}=1 , p is relatively prime to Q and Q is relatively prime to P . Thus, the quantities (P^{-1}) q_i and (Q^{-1}) P_i exist [1 ,L] and therefore P^{-1} A and Q^{-1} B exist.

Accept that the polynomial is a different of , i.e., . At that point, , which implies that . This relates to step 2 of the PRMM calculation, which implies that stride 6 is without lapse since base change in step 3 is sans slip, along these lines PRMM holds. Besides, it must be demonstrated that the subsequent polynomial of Algorithm 4 is a polynomial of degree not as much as n .

$$\deg\{c\} \leq \deg\left\{\frac{v}{Q}\right\} \Rightarrow$$
$$\deg\{c\} \leq \deg\left\{\frac{ab+tp}{Q}\right\} \Rightarrow$$
$$\deg\{c\} \leq \deg\{ab+tp\} - \deg\{Q\} \Rightarrow$$
$$\deg\{c\} \leq \max\{\deg\{ab\}, \deg\{tp\}\} - \deg\{Q\} \Rightarrow$$
$$\deg\{c\} \leq \deg\{tp\} - \deg\{Q\} \Rightarrow$$
$$\deg\{c\} \leq \deg\{Q\} - 1 + n - \deg\{Q\} \Rightarrow$$
$$\deg\{c\} \leq n - 1$$

Since V is the maximum intermediate value of Algorithm 4, its degree must be less than the degree of the polynomial PQ. Under this assumption, we get

$$\deg\{v\} < \deg\{PQ\} \Rightarrow$$
$$\deg\{cQ\} < \deg\{PQ\} \Rightarrow$$
$$\deg\{c\} + \deg\{Q\} < \deg\{P\} + \deg\{Q\} \Rightarrow$$
$$\deg\{c\} < \deg\{P\}$$

Algorithm 3 $GF(p)$ base conversion algorithm (BC)

Input: $\zeta_{\mathcal{B}} = (\zeta_{\mathcal{B}_1}, \zeta_{\mathcal{B}_2}, \cdots, \zeta_{\mathcal{B}_L}), \mathcal{A}, \mathcal{B}$
Output: $\zeta_{\mathcal{A}} = (\zeta_{\mathcal{A}_1}, \zeta_{\mathcal{A}_2}, \cdots, \zeta_{\mathcal{A}_L})$

1: $W_1 \leftarrow 0$
2: $U_1 \leftarrow \zeta_{\mathcal{B}_1}$
3: **for all** $i = 2, \ldots, L$ **do**
4: $\quad U_i \leftarrow \zeta_{\mathcal{B}_i} - \zeta_{\mathcal{B}_1}$
5: $\quad$ **for** $j = 1$ to $i - 1$ **do**
6: $\quad\quad U_i \leftarrow \langle U_i - W_j U_j \rangle_{q_i}$
7: $\quad$ **end for**
8: **end for**
9: $\zeta_{\mathcal{A}_1} \leftarrow 0$
10: **for all** $i = 1, \ldots, L$ **do**
11: $\quad$ **for** $j = 1$ to L **do**
12: $\quad\quad \kappa_{ij} = \langle W_j A_j \rangle_{p_i}$
13: $\quad\quad \zeta_{\mathcal{A}_i} \leftarrow \langle \kappa_{ij} + \zeta_{\mathcal{A}_i} \rangle_{p_i}$
14: $\quad$ **end for**
15: **end for**

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

Algorithm 4 PRNS Montgomery Multiplication (PRMM) in $GF(2^n)$

Input: $a_{\mathcal{T}}, b_{\mathcal{T}}, p_{\mathcal{B}}^{-1}, Q_{\mathcal{A}}^{-1}, p_{\mathcal{A}}$, /*$\deg\{a\} < n$, $\deg\{b\} < n$
Output: $c_{\mathcal{T}}$, /*$\deg\{c\} < n$, $c = abQ^{-1} \mod p$

1: $s_{\mathcal{T}} \leftarrow a_{\mathcal{T}} \cdot b_{\mathcal{T}}$
2: $t_{\mathcal{B}} \leftarrow s_{\mathcal{B}} \cdot p_{\mathcal{B}}^{-1}$
3: $t_{\mathcal{A}} \leftarrow t_{\mathcal{B}}$
4: $u_{\mathcal{A}} \leftarrow t_{\mathcal{A}} \cdot p_{\mathcal{A}}$
5: $v_{\mathcal{A}} \leftarrow s_{\mathcal{A}} + u_{\mathcal{A}}$
6: $c_{\mathcal{A}} \leftarrow v_{\mathcal{A}} \cdot Q_{\mathcal{A}}^{-1}$
7: $c_{\mathcal{B}} \leftarrow c_{\mathcal{A}}$

C. The Proposed Dual-Field Montgomery Multiplication Algorithm

A careful examination of RMM and PRMM algorithms , reveals potential for unification into a common dual-field residue arithmetic Montgomery multiplication (DRAMM) algorithm and a common dual-field base conversion (DBC) algorithm.

The unified algorithms are depicted below as Algorithms 5 and 6, where $\oplus$ represents a dual-field addition/subtraction and $\odot$ represents a dual-field multiplication. An important aspect is that all operations within the DRAMM and the DBC algorithms are now decomposed into simple multiply-accumulate (MAC) operations of word-length r equal to the modulus word length . This allows for a fully-parallel hardware implementation, employing parallel MAC units, each dedicated to a modulus of the RNS/PRNS base.

Algorithm 5 The proposed DRAMM algorithm

Input: $a_{\mathcal{T}}, b_{\mathcal{T}}, \left(-p^{-1}\right)_{\mathcal{B}}, Q_{\mathcal{A}}^{-1}, p_{\mathcal{A}}$, /* $a, b < 2p$
Output: $c_{\mathcal{T}}$, /* $c < 2p$ and $c \equiv abQ^{-1} \mod p$

1: $s_{\mathcal{T}} \leftarrow a_{\mathcal{T}} \odot b_{\mathcal{T}}$
2: $t_{\mathcal{B}} \leftarrow s_{\mathcal{B}} \odot \left(-p^{-1}\right)_{\mathcal{B}}$
3: $t_{\mathcal{A}} \leftarrow t_{\mathcal{B}}$ /* base conversion step
4: $u_{\mathcal{A}} \leftarrow t_{\mathcal{A}} \odot p_{\mathcal{A}}$
5: $v_{\mathcal{A}} \leftarrow s_{\mathcal{A}} \oplus u_{\mathcal{A}}$
6: $c_{\mathcal{A}} \leftarrow v_{\mathcal{A}} \odot Q_{\mathcal{A}}^{-1}$
7: $c_{\mathcal{B}} \leftarrow c_{\mathcal{A}}$ /* base conversion step

Algorithm 6 Dual-field base conversion algorithm (DBC)

Input: $\zeta_{\mathcal{B}} = (\zeta_{\mathcal{B}_1}, \zeta_{\mathcal{B}_2}, \cdots, \zeta_{\mathcal{B}_L}), \mathcal{A}, \mathcal{B}$
Output: $\zeta_{\mathcal{A}} = (\zeta_{\mathcal{A}_1}, \zeta_{\mathcal{A}_2}, \cdots, \zeta_{\mathcal{A}_L})$

1: $W_1 \leftarrow 0$
2: $U_1 \leftarrow \zeta_{\mathcal{B}_1}$
3: **for all** $i = 2, \ldots, L$ **do**
4: $\quad U_i \leftarrow \zeta_{\mathcal{B}_i} \oplus \zeta_{\mathcal{B}_1}$
5: $\quad$ **for** $j = 1$ to $i - 1$ **do**
6: $\quad\quad U_i \leftarrow \langle U_i \oplus W_j \odot U_j \rangle_{q_i}$
7: $\quad$ **end for**
8: **end for**
9: $\zeta_{\mathcal{A}_1} \leftarrow 0$
10: **for all** $i = 1, \ldots, L$ **do**
11: $\quad$ **for** $j = 1$ to L **do**
12: $\quad\quad \kappa_{ij} = \langle W_j \odot U_j \rangle_{p_i}$
13: $\quad\quad \zeta_{\mathcal{A}_i} \leftarrow \langle \kappa_{ij} \oplus \zeta_{\mathcal{A}_i} \rangle_{p_i}$
14: $\quad$ **end for**
15: **end for**

Finally, the conditions from Sections V-A and V-B, for a valid RNS/PRNS transformation of the Montgomery algorithm yield

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

$$\left.\begin{array}{lcl} \deg\{P\} & \geq & n \\ \deg\{Q\} & \geq & n \\ P & \geq & 2p \\ Q & \geq & 4p \end{array}\right\}$$

which means that one should select RNS/PRNS ranges of word length

$$\delta_{\mathcal{A}} \geq \max\{\lceil \log 2p \rceil, n\}$$
$$\delta_{\mathcal{B}} \geq \max\{\lceil \log 4p \rceil, n\}$$

For the bases An and B , individually. Calculations 5 and 6 alongside conditions (12) and (13) shape the complete system for a double field buildup number juggling Montgomery increase. As portrayed some time recently, the structure of the proposed calculation permits it to be reused in the connection of any exponentiation calculation. A conceivable usage is portrayed in Algorithm 7, needing altogether 2n+2 DRAMM increases [4], [46].

D. Conversions

In the following discussion, base A=(P1,P2….PL) shall be used as an example to analyze the conversions to/from residue representations, without loss of generality.

Algorithm 7 Proposed DRAMM modular exponentiation

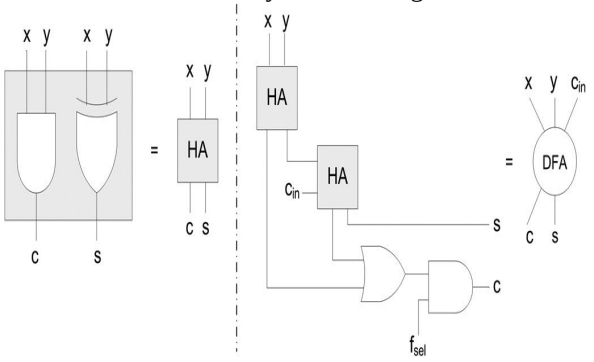
Input: $z_{\mathcal{T}}, e = (e_{n-1} \dots e_1 e_0)_2$
Output: $b_{\mathcal{T}}, b \equiv \langle z^e \rangle_p$

```
b ← 1
for i = n − 1, . . . , 0 do
    b ← DRAMM(b, b)
    if e_i = 1 then
        b ← DRAMM(b, z)
    end if
end for
return b
```

D. Hardware Implementation

a. Dual-Field Addition/Subtraction

A dual-field full-adder (DFA) cell (Fig. 1) is basically a full-adder (FA) cell, equipped with a field-select signal $(fsel)$ that controls the operation mode [33]. In the proposed implementation, 3-level, carry-look ahead adders (CLA) with 4-bit carry-lookahead generator groups (CLG) are employed [47]. An example of a 4-bit dual-field CLA adder is shown in Fig. 4.2. The GAP modules generate the signals $p_i = x_i \mathrm{XOR} y_i$, $g_i = x_i \mathrm{AND} y_i$, $\alpha_i = x_i \mathrm{OR} y_i$, and AND gates along with a signal $fsel$ control whether to eliminate carries or not. The carry-look ahead generator is an network AND- OR based on (19) [47].



$$c_{i+1} = \overset{i}{\underset{j=0}{\mathrm{OR}}} \left(\overset{i}{\underset{k=j+1}{\mathrm{AND}}} \alpha_k \right) g_j \mathrm{OR} \left(\overset{i}{\underset{k=0}{\mathrm{AND}}} \alpha_k \right) c_0$$

Fig. 1 Dual-Field Full-Adder Cell (DFA).

b. Dual-Field Modular/Normal Addition/ Subtraction

With trivial modifications of algorithms for modular addition/subtraction in GF(p) [3], [4], a dual-field modular adder/subtracter (DMAS) shown in Fig. 3 can be mechanized using CLA adders.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

When fsel=0, the circuit is in GF(2^n) mode and the output is derived directly from the top adder which performs a GF(2^n) addition. When fsel=1 , the circuit may operate either as a normal $(2r + \log_2 L)$ -bit adder/subtracter $(conv_mode = 0)$or as a modular adder/subtracter $(conv\_mode = 1)$.
In the first case, the output is the concatenation of the outputs of the two adders. This is required during residue-to-binary conversion, since (18) dictates that ,(2r) -bit quantities need to be added recursively via a normal adder.

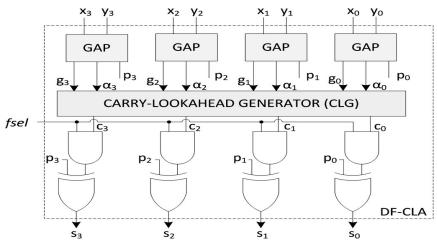


Fig...2 Dual-field CLA.

c. Dual-Field Multiplication

Dual-Field Multiplication A parallel tree 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, i.e.,AND an operation among all operand bits.

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 4x4-bit example of the proposed dual-field multiplier (DM) with output in carry-save format is depicted in Fig. 3

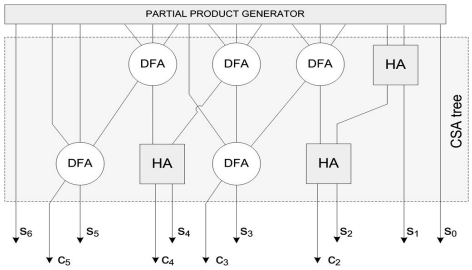


Fig. 3 Dual-field multiplier (DM).

E. MAC Unit

The circuit organization of the proposed MAC unit is shown in Fig.4 . Its operation is analyzed below in three steps, corresponding to the three phases of the calculations it handles, i.e., binary-to-residue conversion, RNS/PRNS Montgomery multiplication, and residue-to-binary conversion.

i. Binary-to-Residue Conversion

Initially, -bit words of the input operands, as implied by (15), are cascaded to each MAC unit and stored in RAM1 at the top of Fig. 4.4. These words serve as the first input to the multiplier, along with the quantities which are stored in a ROM. Their multiplication produces the inner products of (15) or (17) which are added recursively in the DMAS unit. The result is stored via the bus in RAM1. The process is repeated for the second operand and the result is stored in RAM2, so that when the conversion is finished, each MAC unit holds the residue digits of the two operands in the two RAMs. The conversion requires steps to be executed.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

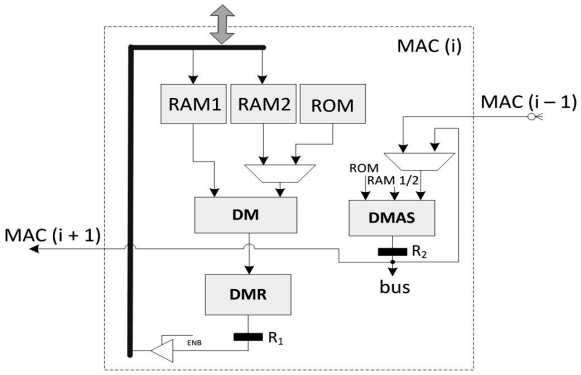


Fig. 4 The proposed MAC unit.

ii. Montgomery Multiplication

The first step of the proposed DRAMM is a modular multiplication of the residue digits of the operands. Since these digits are immediately available by the two RAMs, a modular multiplication is executed and the result in is stored in RAM1 for base and RAM2 for base. Step 2 of DRAMM is a multiplication of the previous result with a constant provided by the ROM. The results correspond to inputs of the DBC algorithm and are stored again in RAM1. All MAC units are updated through the bus with the corresponding RNS digits of all other MACs and a DBC process is initiated.

. Each MAC unit has been assigned to a different color, thus in the overlapped case the color codes signify when a MAC unit performs operations for other units. In the example of Fig..5, MAC(1) handles MAC(4) and MAC(2) handles MAC(3). In each cycle, modular additions and multiplications are performed in parallel in each MAC.

iii. Residue-to-Binary Conversion

Residue-to-binary conversion is essentially repetitions of the DBC algorithm, except for steps 9–14, which are no longer modulo operations. To illustrate the conversion process, assume the generation of the inner products in row 1 of (18). Each product is calculated in parallel in each MAC unit and a "*carry-propagation*" from MAC(1) to is MAC(L) performed to add all inner products. When summation finishes the first digit $z^{(0)}$ f the result is produced in MAC(L) . In parallel with this "*carry-propagation*", the inner products of line 2 are calculated. As soon as a MAC unit completes an addition of carry-propagated inner products for line 1, a new addition for line 2 is performed. The process continues for all lines of (18) and the result is available after steps. The Complete DRAM architecture is depicted in Fig. 8.

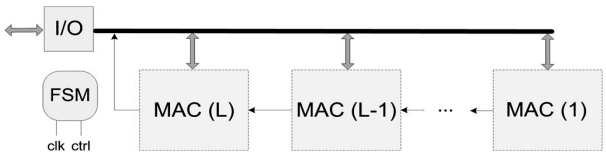


Fig.5 The proposed DRAMM architecture.

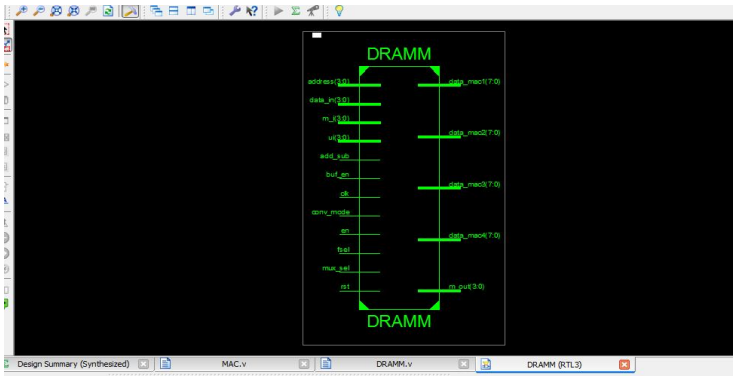
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

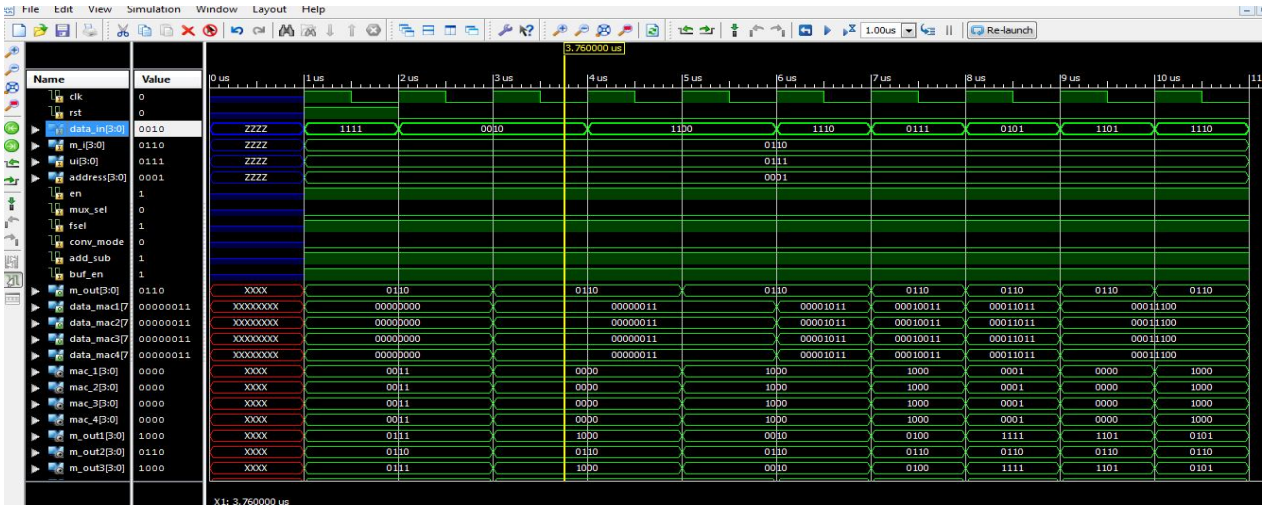
International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2015

IV. SIMULATION RESULTS





V. CONCLUSION

The mathematical framework and a flexible, dual-field, residue arithmetic architecture for Montgomery multiplication in GF(p) and GF(2^n) is developed and the necessary conditions for the system parameters (number of moduli channels, modulus word length) are derived. The proposed DRAMM architecture supports all operations of Montgomery multiplication in GF(p) and GF(2^n) , residue-to-binary and binary-to-residue conversions, MRC for integers and polynomials, dual-field modular exponentiation and inversion, in the same hardware. Generic complexity and real performance comparisons with state-of-the-art works prove the potential of residue arithmetic exploitation in Montgomery multiplication.

REFERENCES

[1] I. Blake, G. Seroussi, andN.Smart, *Elliptic Curves in Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[2] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curves Cryptography*. New York, NY, USA: Springer-Verlag & Hall/CRC, 2004.
[3] J.-P. Deschamps, *Hardware Implementation of Finite-Field Arithmetic*. New York, NY, USA: McGraw-Hill, 2009.
[4] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*. New York, NY, USA: Cambridge Univ. Press, 1986.
[5] 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.
[6] 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.
[7] J.-C. Bajard and L. Imbert, "Brief contributions: A full RNS implementation of RSA," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 769–774, Jun. 2004.