An Efficient Polynomial-Modulus Based Technique for Multiplication of Large Integers

Mustafa Kamal, Magdy Bayoumi, Ashok Kumar
Center for Advanced Computer Studies
University of Louisiana at Lafayette
P.O. Box 44330
Lafayette, LA 70504
{kamal, mab, ak} @cacs.louisiana.edu

***Abstract*—This paper proposes a computational method for polynomial-modulus residue number system (PMRNS), a system in which residue is calculated after decomposing the numbers into polynomial of desired radix. This residue can be used to find product, sum, and difference so that different steps (e.g. finding the residue, addition/subtraction etc. of residues, conversion of result into an integer) can be performed in parallel to enhance the speed of operation. The advantages of the proposed method are: i) it does not need to calculate multiplicative inverse for conversion from the residue product to the number, and ii) it does not need to increase the number of modulus as the number gets larger, instead radix can be increased to meet the computational need.**

I. INTRODUCTION

It is known that it is possible to uniquely determine a nonnegative integer given its residues with respect to a set of moduli, provided that the integer is known to be smaller than the product of the moduli. This can be accomplished by Chinese remainder theorem (CRT)

The obtained advantage is that by considering the residue of large integers modulo a set of moduli, these large integers are decomposed into smaller ones that can easily be added, subtracted and multiplied. These operations are performed in a carry free fashion implying speed and parallelism.

In a ring of polynomials over any field, there again is a Chinese remainder theorem, here Polynomial multiplications can be performed in the same way as integer, by taking polynomial residues, large polynomials can be decomposed into smaller one that are easy to multiply, add and subtract. Following two theorems describes the above concepts.

Theorem.1: *Given a set of polynomials* $m^{(0)}(x), m^{(1)}(x), \ldots\ldots\ldots, m^{(n)}(x)$ *that are pair wise relatively prime and a set of polynomials* $c^{(0)}(x), c^{(1)}(x), \ldots\ldots\ldots, c^{(x)}$ *with deg* $c^{(i)}(x)$ > *deg* $m^{(i)}(x)$ *then the system of equations* $c^{(i)}(x) = c(x)$ *(mod* $m^{(i)}(x)$*)* $i = 0, \ldots\ldots, k$ *has at most one solution for* $c(x)$ *satisfying* $\deg c(x) < \sum_{i=0}^{k} \mathbf{deg}\ m^{(i)}(x)$ *[3].*

Theorem.2: *Let* $M(x) = \prod_{r=0}^{k} m^{(r)}(x)$ *be a product of relatively prime polynomials; let* $M^{(i)}(x) = M(x) \big/ m^{(i)}(x)$ *and* $N^{(i)}(x)$ *satisfy* $N^{(i)}(x)M^{(i)}(x) + n^{(i)}(x)m^{(i)}(x)(x = 1)$. *Then the system of congruences be a product of relatively prime polynomials; let* $c^{(i)}(x) = c(x)\left(\mathrm{mod}\ m^{(i)}(x)\right)$ $i = 0 \ldots k$ *is uniquely solved by*

$$c(x) = \sum_{i=0}^{k} c^{(i)}(x)N^{(i)}(x)M^{(i)}(x) \quad (\mathrm{mod}\ M(x)) \qquad [3]$$

Specific application of the above two theorems by Skavantozos and Taylor[2], results in the development of Polynomial Residue Number System(PRNS).

The Polynomial Residue Number System (PRNS) examines the problem of multiplying two *(N-1)* st-degree polynomials $mod(x^n \pm 1)$, over some modular ring $z_m = \{0,1, \ldots m-1\}$, a ring which is closed with respect to the operations of addition and multiplication ***mod*** *m* [2].

0-7803-9197-7/05/$20.00 © 2005 IEEE.

First known modular polynomial basis multiplier over $GF(2^m)$ was proposed by Mastrovito[4] and extensive research has been made on this algorithm using different set of polynomials and many different formulation of Mastrovito algorithm has been proposed.

Sunar and Koc[5] mdified the Mastrovito algorithm using trinomials and have shown the m^2-1 XOR and m^2 AND gates are sufficient to implement the multiplier. Halbutogullari and Koc[6] developed a method for constructing Mastrovito multiplier for arbitrary irreducible polynomial, trinomials, all-one polynomials and equally spaced polynomials. So far for these special polynomials, XOR gate count and time delay of Halbutogullari-Koc algorithm appear to be lowest. Comparison of complexities of Halbutogullari-Koc and Mastrovito for related s-ESP-based multipliers are shown Table I.

Best known time complexity for multiplying large integers using software has been developed by Dan Zuras[7] having time complexity $O(n^{1.365})$.

To satisfy the high speed requirements of multiplier, proposed technique introduces a new concept of multiplying large integer numbers, decomposing it into smaller one and perform the arithmetic operation and also the range change can be done without increasing the number of moduli, thereby increasing the speed. This technique addresses not only conversion into residue and multiplication but also inversion process.

II. BASIC DEFINITIONS.

Definitions1: Monic Polynomial is a polynomial whose coefficient f_n with largest index is equal to 1.

Definitions2: Remainder Theorem states that when a polynomial $f(x)$ is divided by $(x-d_i)$ in $F(x)$ is $f(d_i)$.

III. PROPOSED TECHNIQUE.

Step 1: Integer to polynomial conversion.

Conversion of integer number into polynomial of desired radix. All numbers in computer are stored in binary form i.e in radix 2. Now to change the radix from 2 to 2^n, the n digits are taken at a time from the least significant position to the most significant position.

Step2: Calculation of residue:

If we choose the root of the modulus to 0 or 2^n, the residue conversion is only a shift and add operation. This is done by conversion matrix such that all residue can be calculated in parallel.

Step 3: Multiplication of residues.

Most efficient multiplier for small numbers can be used.

Step 4: Inversion of polynomial product into number.

This is obtained by multiplying polynomial product by the inversion matrix. This also can be done in parallel.

IV. SELECTION OF POLYNOMIAL MODULUS AND RESIDUE COMPUTATION.

PMRNS performs the mathematical operations by expanding the numbers into polynomial, such that the indeterminate x of the polynomial $A_r(x) = a_0 + a_1x + a_2x^2 + \ldots\ldots + a_{n-1}x^{n-1}$ is the radix *r* can be any integer. By increasing the value of *r* we can reduce the degree of polynomial.

Example 1: Let A = 570 and if radix r = 10 then $A_{10} = 2 + 7.10 + 5.10^2$ can be represented as polynomial $A_{10}(x) = 2 + 7x + 5x^2$ having degree of 2. Now we can reduce the degree of the same number if we choose r = 100 generating polynomial $A_{100}(x) = 72 + 5x$ and degree of 1.

Each modulus should be monic polynomial of degree one. From Chinese remainder theorem we know that the solution of *n-tuple* residue set is unique if the number represented by *n-tuple residue set is* less than the product of the modulus $M = m_1.m_2.m_3\ldots\ldots m_d$.

If we choose a set of polynomial modulus $m_1(x) = (x-d_1)$, $m_2(x) = (x-d_2)$, $m_3(x) = (x-d_3)$, …. $m_k(x) = (x-d_k)$ then the product of modulus is $M(x) = (x-d_1)(x-d_2)(x-d_3)\ldots(x-d_k) = p_0 + p_1x + p_3x + \ldots\ldots p_kx^k$. It is obvious the solution of the *n-tuple* polynomial residue set will be unique *if the degree of the polynomial represented by the product of the polynomial modulus* $M(x)$ *is greater than the degree of* $A_r(x)$ otherwise the number cannot be represented uniquely. Careful choice of the d_i can reduce the magnitude of the residue.

Example 2: Using remainder theorem it is very easy to calculate the remainder of the polynomial. If $m_1(x) = x-1$ and $A_{10}(x) = 2 + 7x + 5x^2$ then residue $R_1 = 2 + 7.1 + 5.1^2 = 14$.

V. *PMRNS* THEOREM AND PROOF.

$$A = \sum_{i=1}^{k} \frac{\overline{m}}{l_i} R_i \quad \text{where } \overline{m_i} = \prod_{j=1 \,\&\, i \neq j}^{k} (r - d_j), \quad l_i = \prod_{j=1 \,\&\, i \neq j}^{k} (d_i - d_j)$$

and the product of modulus $M(x) = (x-d_1)(x-d_2)\ldots\ldots(x-d_k)$.

Proof: The proof can be obtained by mathematical induction and it is omitted for brevity. An example is given for illustration next. For simplicity let us assume $A(x) = a_0 + a_1 x$ and $M(x) = (x - d_1)(x - d_2)$. Here the degree of $M(x)$ is *k=2*. Now $R_1 = A(d_1) = a_0 + a_1 d_1$, $R_2 = A(d2) = a_0 + a_1 d_2$, $\overline{m_1} = (r - d_2)$, $\overline{m_2} = (r - d_1)$, $l_1 = (d_1 - d_2)$ and $l_2 = (d_2 - d_1)$. Therefore $A = \frac{(r-d_2)}{(d_1-d_2)}(a_0 + a_1 d_1) + \frac{(r-d_1)}{(d_2-d_1)}(a_0 + a_1 d_2) = \frac{(a_0 + a_1 r)(d_1 - d_2)}{(d_1 - d_2)} = (a_0 + a_1 r)$.

Example 3: If $A = 115$, $B = 308$ and $r = 8$ then $A = 3 + 6.8 + 1.8^2$ and $B = 4 + 6.8 + 4.8^2$. Therefore, $A(x) = 3 + 6x + x^2$ and $B(x) = 4 + 6x + 4x^2$. Let $M(x) = (x+2)(x+1)x(x-1)(x-2)$ then $R(A) = \{-5,-2,3,10,19)$ and $R(B) = \{8,2,4,14,32\}$. The product $R(A).R(B) = \{-40,-4,12,140,608\}$. Applying PMRNS theorem we get $A.B = \frac{9.8.7.6}{-1.-2.-3.-4}(40) + \frac{10.8.7.6}{1.-1.-2.-3}.(-4) + \frac{10.9.7.6}{2.1.-1.-2}.(12) + \frac{10.9.8.6}{3.2.1.-1}.(140) + \frac{10.9.8.7}{4.3.2.1}(608) = 35420$.

VI. A METHOD FOR PARALLEL COMPUTATION.

Maximum speed could be achieved by parallel calculation of residues and the co-efficient of the polynomial, this is done by developing three matrix $[D]$, $[a]$ and $[D]^{-1}$.

Where $[D] = \begin{bmatrix} d_0^0 & d_1^0 & d_2^0 & . & d_k^0 \\ d_0^1 & d_1^1 & d_2^1 & . & d_k^1 \\ d_0^2 & d_1^2 & d_2^3 & . & d_k^3 \\ . & . & . & . & . \\ d_0^k & d_1^k & d_2^k & . & d_k^k \end{bmatrix}$.

$[a]$ is a row matrix made by the co-efficients of the polynomial as $[a] = [a_0 \quad a_1 \quad a_2 \quad . \quad a_k]$ and $[D]^{-1}$ matrix is the inverse of the matrix $[D]$.

After inverting $[D]$, we may find some of the elements are fraction. To avoid fraction multiplication, multiply each element of the $[D]^{-1}$ by the l.c.m L (least common multiple) of the denominator of each element and generate a matrix $\overline{[D]}^{-1}$ then divide by L to get $\frac{1}{L}\overline{[D]}^{-1}$.

Now if we multiply matrix $[a]$ and matrix $[D]$ we get the set of residue $\{R\}$. After the required mathematical operation we can convert those residues to the co-efficients of the polynomial just multiplying by the matrix $\frac{1}{L}\overline{[D]}^{-1}$.

Example 4: Multiplication of two numbers A=115 and B =308.

Let *r=16*, $m_1(x) = (x+2)$, $m_2(x) = (x+1)$, $m_3(x) = x$, $m_4(x) = (x-1)$, $m_5(x) = (x-2)$ and $M(x) = (x+2)(x+1)x(x-1)(x-2)$ then $A = 3 + 7.16$ and $B = 4 + 3.16 + 1.16^2$ can be represented by $A(x) = 3 + 7x$ and $B(x) = 4 + 3x + x^2$.

$$[D] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{bmatrix}, [D]^{-1} = \begin{bmatrix} 0 & \frac{1}{12} & -\frac{1}{24} & -\frac{1}{12} & \frac{1}{24} \\ 0 & -\frac{1}{3} & \frac{2}{3} & \frac{1}{6} & -\frac{1}{6} \\ 1 & 0 & -\frac{5}{4} & 0 & \frac{1}{4} \\ 0 & \frac{2}{3} & \frac{2}{3} & -\frac{1}{6} & \frac{1}{6} \\ 0 & -\frac{1}{12} & -\frac{1}{24} & \frac{1}{12} & \frac{1}{24} \end{bmatrix}.$$

Here lcm = 24 therefore $\overline{[D]}^{-1} = \begin{bmatrix} 0 & 2 & -1 & -2 & 1 \\ 0 & -16 & 16 & 4 & -4 \\ 24 & 0 & -30 & 0 & 6 \\ 0 & 16 & 16 & -4 & -4 \\ 0 & -2 & -1 & 2 & 1 \end{bmatrix}$

$$[I_n] = \frac{1}{L}\overline{[D]}^{-1} = \frac{1}{24}\begin{bmatrix} 0 & 2 & -1 & -2 & 1 \\ 0 & -16 & 16 & 4 & -4 \\ 24 & 0 & -30 & 0 & 6 \\ 0 & 16 & 16 & -4 & -4 \\ 0 & -2 & -1 & 2 & 1 \end{bmatrix}$$

$[a] = [3 \quad 7 \quad 0 \quad 0 \quad 0]$ and $[b] = [4 \quad 3 \quad 1 \quad 0 \quad 0]$

$R(A) = [a][D] = [-11 \quad -4 \quad 3 \quad 10 \quad 17]$

$R(B) = [b][D] = [2 \quad 2 \quad 4 \quad 8 \quad 14]$

$R(A).R(B) = [R_p][-22 \quad -8 \quad 12 \quad 80 \quad 238]$

$A.B = [R_p][I_n] = [12 \quad 37 \quad 24 \quad 7 \quad 0] = 12 + 37.16 + 24.16^2 + 7.16^3 = 35420$.

The limit of maximum number can be increased without any alteration of $M(x)$ but just by increasing the radix *r*.

Example 5: If $A = 308$, $B = 4627$ and radix *r=16* then $A(x) = 4 + 3x + x^2$ and $B(x) = 3 + x + 2x^2 + x^3$. The degree of the polynomial of the product $A(x).B(x)$ is 5 which is equal to the degree of $M(x)$, therefore, we cannot get unique result. Thus increasing the radix *r=32* we get $A(x) = 20 + 9x$ and $B(x) = 19 + 16x + 4x^2$ results in 3 as the degree of the polynomial of the product $A(x).B(x)$ which is less than the degree of $M(x)$.

VII. CONCLUSIONS.

PMRNS performs the residue conversion, multiplication of residues and inversion from the product of the residue simultaneously in a pipeline. Similar research is scarce to find for comparison. Best known complexity of multiplying large integers is in the range from $O(N^{1.465})$ to $O(N^{1.365})$ [7], depending on length of the number. This research leads to this method which finds the complexity of multiplying to integers in the range from $O(N^{1.5})$ to $O(N^{1.24})$ as shown in the Table 2. This method also has the following advantages.

1. Limit of the maximum number does not depend on the value of $M(x)$ but on the degree of the product of the polynomial-modulus.

2. Any number can be used as a radix.

Above two criteria provides PMRNS highly flexible in increasing the maximum range of the number. Also VLSI implementation requires less space.

3. To accommodate a larger number, each time we do not have to increase the degree of the polynomial, just by increasing the radix to a higher number during the initial conversion will do the job.

4. By appropriate selection of the set d_i we can make most of the element of the matrix $[D]$ and inverse of $[D]$ as power of 2 such that residue and coefficient calculation will be only shift and addition operations.

5. In the computer, numbers are represented in binary. If we select *r* as power of 2 then the calculation of the co-efficient of the polynomial is instantaneous.

Therefore, PMRNS provides much faster residue conversion than RNS system because RNS needs division for getting residue.

6. This system splits a multiplication into several small multiplications such that all of them can be done in parallel thereby farther reducing the total computation time.

ACKNOWLEDGMENT

The authors acknowledge the support of the U.S. Department of Energy (DoE), EETAPP program DE97ER12220 and the Governor's Information Technology Initiative.

REFERENCES

[1] Neil M. Wigley and Graham A. Jullien, "On Modulus Replication for Residue Arithmetic Computaions of complex Inner Products," IEEE Trans. On Computer VOL 39, No 8, August 1990.

[2] Alexander Skavantozox and Frd J. Taylor, " On the polynomial Residue Number System,"IEEE Trans. On Signal processing, Vol. 39, No 2, Feb 1991.

[3] Richard E. Blahut, "Fast Algorithms for Digital Signal Processing,"Addision-Wesley Publishign Company.

[4] E.D Mastrovito, "VLSI Designs for Multiplication over Finite Fields $GF(2^m)$,"Proc. Sixth Symp. Applied Algebra, Algebric Algorithms, and Error Correcting Codes (AAECC-6) pp 297-309.

[5] B. Sunar and C.K. Koc,"Mastrovito Multiplier for All Trinomials,"IEEE Trans. Computers, vol. 48, no. 5 pp. 522-527.

[6] A. Halbutogullari and C.K. Koc. ,"Mastrovito Multiplier for General Irreducible Polynomials," IEEE Trans. Computers vol 49, no. 5, pp. 503, May 2000.

[7] Dan Zuras, "More On Squaring and Multiplying Large Integers," IEEE Trans. On Computers Vol. 43m No 8, August 1994.

Table 1

Comparison of Related s-ESP-Based polynomial Basis multiplier

Reference	**#AND**	**#XOR**	**Time delay**
Mastrovito[4]	m^2	$\frac{2s+1}{2s}m^2 - \frac{3}{2}m$	$T_A + (1 + \lceil \log_2 m \rceil)T_x$
Halbutogullari-Koc[6]	m^2	$m^2 - s$	$T_A + (1 + \lceil \log_2 m \rceil)T_x$

Table 2

For N=1024

Regular Multipliction	**PMRNS**			
Complixity	Run Time	Bit-Time	#Modulus	Precision
$O(N^2)$	$N^{1.5}$	32768	3	Double
	$N^{1.4}$	16384	3	Single
	$N^{1.38}$	14263	5	Double
	$N^{1.24}$	5405	5	Single