
PERGAMON Computers and Mathematics with Applications 39 (2000) 227–238

An International Journal
computers & mathematics
with applications

www.elsevier.nl/locate/camwa

Arithmetic Division in RNS Using Galois Field $GF(p)$

S. TALAHMEH AND P. SIY
Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202, U.S.A.
<asalm><psiy>@ece.eng.wayne.edu

(Received March 1998; revised and accepted February 1999)

Abstract—This paper develops an enhanced algorithm for the arithmetic division problem in the Residue Number System. The proposed algorithm is based on Galois Field Theory $GF(p)$. Mapping the arithmetic division problem over the Galois Field $GF(p)$ eliminates many of the limitations of existing algorithms. The advantage of the proposed algorithm is that it has no restriction on the dividend and the divisor, no mixed radix conversion, no quotient estimation before division, no reciprocal estimation of the divisor, and no based extension operation.

Keywords—Computer arithmetic, Modular arithmetic, Galois field, Number theory, Parallel computation.

1. INTRODUCTION

The arithmetic division in residue number system RNS is usually classified into three categories: Division Remainder Zero (DRZ), scaling, and general division [1]. DRZ and scaling in RNS have limited applications. Different algorithms have been developed to speed up scaling [2–4]. The general division problem in RNS has attracted the attention on many researchers to design high-speed multimoduli ALU systems. Digital systems that are built around RNS arithmetic units may play an important role in high-speed real time systems that support parallel processing of integer-valued data [5]. Addition, subtraction, and multiplication operations, called modular operations, can be performed very fast without carry or borrow propagation [1,6]. The nonmodular operations division, magnitude comparison, sign detection, and overflow detection are still relatively slow [6,7]. Any speed-up algorithm for such slow operations will dramatically improve the performance of multimoduli ALU systems.

Several algorithms for RNS general division have been proposed in the past. These algorithms can be classified into two groups: multiplicative and subtractive [6,8]. Most of multiplicative algorithms first compute (or estimate) the reciprocal of the divisor, and then the reciprocal is multiplied by the dividend. The subtractive algorithms employ subtraction of multiples of the divisor from the dividend until the difference becomes less than the divisor. There are several RNS division algorithms that are classified as multiplicative algorithms, among these [9–12]. All these division algorithms use Mixed Radix Conversion (MRC) to find (or estimate) the reciprocal of the divisor and to compare numbers. These MRC-based algorithms are generally slow and require

0898-1221/00/$ - see front matter Typeset by AMS-TEX
PII: S0898-1221(00)00056-0

a lot of arithmetic computations. On the other hand, there are several subtractive algorithms presented in literature, among these [13,14]. These algorithms do not require MRC computations, but some nonmodular computations are needed [2,3]. The subtractive algorithm presented in [14] seems particularly attractive because an efficient parity checking method was used for comparison, sign, and overflow detection. Most of the existing division algorithms have drawbacks that make them less suitable as solutions of the RNS division problem.

In this paper, we present a very fast algorithm for the general division problem in RNS using index mapping over $GF(p)$. The enhanced algorithm has the following properties: very fast compared to published algorithms, no restriction on the dividend and the divisor (except zero divisor), no quotient estimation before the division, no reciprocal estimation of the divisor, and no base extension operation.

In decimal arithmetic, logarithms are frequently used for multiplication and division. In RNS, an analogous method is used called index calculus [1]. Using index transformation over the Galois Field $GF(p)$, multiplication and division operations can be implemented by addition and subtraction, respectively. Multiplication operation is a modular operation, therefore, multiplication can be done as addition in RNS. In terms of hardware implementation, addition in RNS is easier than multiplication [14]. Division, however, is one of the nonmodular operations in RNS, therefore, the proposed index transformation over $GF(p)$ will definitely improve the computational speed and hardware cost.

2. GALOIS FIELD $GF(p)$

Finite Galois fields are of two types: prime fields $GF(p)$ and polynomial fields $GF(p^q)$, where p is a prime number, and q is a positive integer. All Galois fields have the property that all the nonzero elements can be generated by using a primitive root (element) denoted here by g. This property can be exploited in doing exact division over the Galois field $GF(p)$. This property is defined as follows [15].

DEFINITION 1. *Let p be any prime number, and let g be any primitive root of p, then to each integer a, relatively prime to p, there is a unique integer i, denoted as $i = \operatorname{ind}_g a$, such that*

$$a = \left|g^i\right|_p, \qquad 0 \le i < p-1. \tag{1}$$

Indices over Galois field $GF(p)$ possess the following important properties:

(1) $\operatorname{ind}_g 1 = 0$,
(2) $\operatorname{ind}_g(a \cdot b) = |\operatorname{ind}_g a + \operatorname{ind}_g b|_{p-1}$,
(3) $\operatorname{ind}_g a^n = |n \cdot \operatorname{ind}_g a|_{p-1}$,
(4) $\operatorname{ind}_g a = |\operatorname{ind}_g g' + \operatorname{ind}_g a|_{p-1}$, where g' is any other primitive root.

In some cases, the sum of indices may exceed the highest index value of $GF(p)$. In this case, Fermat's theorem may be used.

THEOREM 1. Fermat theorem. *If p is prime, then*

$$\left|a^p\right|_p = |a|_p, \qquad \text{for all integers } a. \tag{2}$$

LEMMA 1. *If p is prime and a is an integer, then*

$$\left|a^{p-1}\right|_p = 1. \tag{3}$$

PROOF. From equation (2),

$$\left|a^p\right|_p = \left|a^{p-1} \cdot a\right|_p = \left|\left|a^{p-1}\right|_p \cdot |a|_p\right|_p = |a|_p.$$

Therefore, $|a^{p-1}|_p$ must be equal to 1, which proves Lemma 1.

LEMMA 2. *If p is a prime number, and n and a are integers, then*

$$|a^n|_p = \left|a^{|n|_{p-1}}\right|_p. \tag{4}$$

PROOF. For any integer n, we can write

$$n = \left\lfloor \frac{n}{p-1} \right\rfloor (p-1) + |n|_{p-1}. \tag{5}$$

Consequently,

$$|a^n|_p = \left|a^{\lfloor n/(p-1)\rfloor (p-1)} \cdot a^{|n|_{p-1}}\right|_p. \tag{6}$$

By Lemma 1, the term $|a^{\lfloor n/p-1\rfloor (p-1)}|_p$ is equal to 1, hence,

$$|a^n|_p = \left|a^{|n|_{p-1}}\right|_p. \tag{7}$$

3. THE DIVISION PROBLEM

In RNS, the arithmetic division problem is classified into three separate categories:

(1) Division-Remainder-Zero (DRZ),
(2) scaling,
(3) general division.

Categories (1) and (2) are special cases with certain applications. DRZ is obviously of restricted use, since it must be known *a priori* whether the remainder is zero or not. Scaling is division with fixed divisors. In our case, the divisor is restricted to a product of some moduli. General division, therefore, is used when it is not known *a priori* that the dividend is a multiple of the divisor or when it is not known that the divisor is one of the permissible divisors required by scaling.

3.1. DRZ Algorithm over Galois Field $GF(p)$

DRZ refers to the calculation of a quotient $\lfloor x/y \rfloor$ when it is known *a priori* that the remainder is zero. Galois field $GF(p)$ can be used to solve very efficiently the arithmetic DRZ problem, as shown in the following theorems.

THEOREM 2. *Let $\{q_n\} = \{1, 2, \ldots, p-1\}$ be a DRZ group and $\{i_n\} = \{0, 1, \ldots, p-2\}$ its associative isomorphic subtractive group with the mapping defined by $q_n = |g^{i_n}|_p$, where g is a primitive element of $GF(p)$. Let q_x and q_y be two integer numbers such that q_y divides q_x. Thus, the DRZ division $|q_x/q_y|_p$ can be defined as*

$$\left|\frac{q_x}{q_y}\right|_p = \left|g^{|i_x - i_y|_{p-1}}\right|_p = \left|g^{|i_x + (p-1-i_y)|_{p-1}}\right|_p. \tag{8}$$

PROOF. The proof follows from the Galois mapping and uniqueness of the indices. Since q_y divides q_x without remainder, then by (1),

$$\begin{gathered} q_x = |g^{i_x}|_p \qquad \text{and} \qquad q_y = |g^{i_y}|_p, \\ \left|\frac{q_x}{q_y}\right|_p = \left|\frac{|g^{i_x}|_p}{|g^{i_y}|_p}\right|_p = \left|\frac{g^{i_x}}{g^{i_y}}\right|_p. \end{gathered} \tag{9}$$

The indices i_x and i_y are unique integers, and therefore,

$$\left|\frac{q_x}{q_y}\right|_p = \left|g^{(i_x - i_y)}\right|_p . \tag{10}$$

By Lemma 2,

$$\left|\frac{q_x}{q_y}\right|_p = \left|g^{|i_x - i_y|_{p-1}}\right|_p . \tag{11}$$

Taking the additive inverse of $-i_y$, we get

$$\left|\frac{q_x}{q_y}\right|_p = \left|g^{|i_x - i_y|_{p-1}}\right|_p = \left|g^{|i_x + (p-1-i_y)|_{p-1}}\right|_p .$$

In RNS systems, numbers are represented in several residues depending on the set of moduli used in the system. This permits the modulo subtractor (or adder) to be split into a number of smaller (and so usually faster) subtractors, thereby decreasing the total area, decreasing the word length of each channel, and increasing the speed. This approach is based on Theorem 3. ∎

THEOREM 3. *Let $\{q_n\} = \{1, 2, \ldots, p-1\}$ be a DRZ division group, and $\{i_n\} = \{0, 1, \ldots, p-2\}$ is its associative isomorphic subtractive group over a prime p and primitive root g. If $p - 1 = \prod_{j=0}^{r-1} m_j$ such that $(m_0, m_1, \ldots, m_{r-1})$ are relatively prime, $\operatorname{ind}_g x = (x_0, x_1, \ldots, x_{r-1})$ and $\operatorname{ind}_g y = (y_0, y_1, \ldots, y_{r-1})$ and y divides x, the arithmetic division $|x/y|_p$ can be defined as*

$$\left|\frac{x}{y}\right|_p = \left|g^{\left(|x_0 - y_0|_{m_0}, |x_1 - y_1|_{m_1}, \ldots, |x_{r-1} - y_{r-1}|_{m_{r-1}}\right)}\right|_p . \tag{12}$$

PROOF. Both $\operatorname{ind}_g x = i_x$, $\operatorname{ind}_g y = i_y$, are members of the subtractive group $\{i_n\}$. Since $p - 1 = \prod_{j=0}^{r-1} m_j$ and $GCD(m_i, m_j) = 1$ for $i \neq j$, there is a unique RNS representation for each number $k \in \{i_n\}$. Consequently,

$$\begin{aligned}
(i_x)_{\text{RNS}} &= \left(|i_x|_{m_0}, |i_x|_{m_1}, \ldots, |i_x|_{m_{r-1}}\right) = (x_0, x_1, \ldots, x_{r-1}) , \\
(i_y)_{\text{RNS}} &= \left(|i_y|_{m_0}, |i_y|_{m_1}, \ldots, |i_y|_{m_{r-1}}\right) = (y_0, y_1, \ldots, y_{r-1}) , \\
(i_x - i_y)_{\text{RNS}} &= \left(|x_0 - y_0|_{m_0}, |x_1 - y_1|_{m_1}, \ldots, |x_{r-1} - y_{r-1}|_{m_{r-1}}\right) .
\end{aligned}$$

Since $p - 1 = \prod_{j=0}^{r-1} m_j$, then $|i_x - i_y|_{p-1} = (|x_0 - y_0|_{m_0}, |x_1 - y_1|_{m_1}, \ldots, |x_{r-1} - y_{r-1}|_{m_{r-1}})$. Using Theorem 2, we get

$$\left|\frac{x}{y}\right|_p = \left|g^{|i_x - i_y|_{p-1}}\right|_p = \left|g^{\left(|x_0 - y_0|_{m_0}, |x_1 - y_1|_{m_1}, \ldots, |x_{r-1} - y_{r-1}|_{m_{r-1}}\right)}\right|_p . \qquad \blacksquare$$

EXAMPLE 1. Consider a prime number $p = 43$ and $g = 3$. The DRZ division group $\{q_n\} = \{1, 2, \ldots, 42\}$, and its associative subtractive isomorphic group $\{i_n\} = \{0, 1, \ldots, 41\}$. Since $p-1 = 42$, the proper moduli are $m_0 = 2$, $m_1 = 3$, and $m_2 = 7$. Each and every integer i_n can be uniquely represented as 3-tuple $r_n = (|i_n|_2, |i_n|_3, |i_n|_7)$. The set of all 3-tuples form the group $\{r_n\}$. The isomorphic mappings between the $\{q_n\}$ and $\{r_n\}$ groups are illustrated in Table 1. Table 2 shows the corresponding inverse mapping over $GF(43)$. The intermediate index i_n is not shown, since it is not used in the table look-up implementation. The generation of table entries will be illustrated for $q_n = 27$. The index is first calculated as follows:

$$i_n = \operatorname{ind}_3 27 = 3, \qquad \text{since } 27 = \left|3^{i_n}\right|_{43} = \left|3^3\right|_{43} .$$

Table 1. Look-up table for $GF(43)$, $p = 43$, $g = 3$, $m_0 = 2$, $m_1 = 3$, $m_2 = 7$.

ADD $= q_n$	r_n	ADD $= q_n$	r_n	ADD $= q_n$	r_n	ADD $= q_n$	r_n	ADD $= q_n$	r_n
1	(0,0,0) 0 00 000	10	(0,1,3) 0 01 011	19	(1,1,6) 1 01 110	28	(1,2,5) 1 10 101	37	(1,1,0) 1 01 000
2	(1,0,6) 1 00 110	11	(0,0,2) 0 00 010	20	(1,1,2) 1 01 010	29	(1,2,6) 1 10 110	38	(0,1,4) 0 01 100
3	(1,1,1) 1 01 001	12	(1,1,5) 1 01 101	21	(0,0,1) 0 00 001	30	(1,2,4) 1 10 100	39	(1,0,5) 1 00 101
4	(0,0,5) 0 00 101	13	(0,2,4) 0 10 100	22	(1,0,1) 1 00 001	31	(0,1,6) 0 01 110	40	(0,1,1) 0 01 001
5	(1,1,4) 1 01 100	14	(0,2,6) 0 10 110	23	(0,1,2) 0 01 010	32	(1,0,2) 1 00 010	41	(0,0,6) 0 00 110
6	(0,1,0) 0 01 000	15	(0,2,5) 0 10 101	24	(0,1,5) 0 01 101	33	(1,1,3) 1 01 011	42	(1,0,0) 1 00 000
7	(1,2,0) 1 10 000	16	(0,0,3) 0 00 011	25	(0,2,1) 0 10 001	34	(1,2,2) 1 10 010		
8	(1,0,4) 1 00 100	17	(0,2,3) 0 10 011	26	(1,2,3) 1 10 011	35	(0,0,4) 0 00 100		
9	(0,2,2) 0 10 010	18	(1,2,1) 1 10 001	27	(1,0,3) 1 00 011	36	(0,2,0) 0 10 000		

Table 2. Inverse mapping table of $GF(43)$, $p = 43$, $g = 3$, $m_0 = 2$, $m_1 = 3$, $m_2 = 7$.

ADD $= r_n$	q_n	ADD $= r_n$	q_n	ADD $= r_n$	q_n	ADD $= r_n$	q_n	ADD $= r_n$	q_n
(0,0,0) 0 00 000 = 0	1	(0,1,2) 0 01 010 = 10	23	(0,2,4) 0 10 100 = 20	13	(1,0,6) 1 00 110 = 38	2	(1,2,1) 1 10 001 = 49	18
(0,0,1) 0 00 001 = 1	21	(0,1,3) 0 01 011 = 11	10	(0,2,5) 0 10 101 = 21	15	(1,1,0) 1 01 000 = 40	37	(1,2,2) 1 10 010 = 50	34
(0,0,2) 0 00 010 = 2	11	(0,1,4) 0 01 100 = 12	38	(0,2,6) 0 10 110 = 22	14	(1,1,1) 1 01 001 = 41	3	(1,2,3) 1 10 011 = 51	26
(0,0,3) 0 00 011 = 3	16	(0,1,5) 0 01 101 = 13	24	(1,0,0) 1 00 000 = 32	42	(1,1,2) 1 01 010 = 42	20	(1,2,4) 1 10 100 = 52	30
(0,0,4) 0 00 100 = 4	35	(0,1,6) 0 01 110 = 14	31	(1,0,1) 1 00 001 = 33	22	(1,1,3) 1 01 011 = 43	33	(1,2,5) 1 10 101 = 53	28
(0,0,5) 0 00 101 = 5	4	(0,2,0) 0 10 000 = 16	36	(1,0,2) 1 00 010 = 34	32	(1,1,4) 1 01 100 = 44	5	(1,2,6) 1 10 110 = 54	29
(0,0,6) 0 00 110 = 6	41	(0,2,1) 0 10 001 = 17	25	(1,0,3) 1 00 011 = 35	27	(1,1,5) 1 01 101 = 45	12	–	
(0,1,0) 0 01 000 = 8	6	(0,2,2) 0 10 010 = 18	9	(1,0,4) 1 00 100 = 36	8	(1,1,6) 1 01 110 = 46	19	–	
(0,1,1) 0 01 001 = 9	40	(0,2,3) 0 10 011 = 19	17	(1,0,5) 1 00 101 = 37	39	(1,2,0) 1 10 000 = 48	7	–	

The 3-tuples is then obtained from i_n,

$$r_n = (|i_n|_2, |i_n|_3, |i_n|_7) = (|3|_2, |3|_3, |3|_7) = (1, 0, 3).$$

That is, at ROM address ADD= $q_n = 27$, the entry is $(1, 0, 3) = (1\,00\,011)_2$. The result is stored as binary word consisting of 1-bit, 2-bit, and 3-bit corresponding to mod 2, mod 3, and mod 7 residue, respectively.

From Theorem 3, it follows that subtraction modulo p can be done as a set of concurrent subtraction (or addition) operations on a number of smaller relatively prime moduli, such that

$$p - 1 = \prod_{j=0}^{r-1} m_j.$$

A number of choices exist for the selection of the set of moduli for the RNS systems. To increase speed and efficiency of DRZ division, the moduli should be selected based on the following criteria.

(a) $p - 1 = \prod_{j=0}^{r-1} m_j$.
(b) They must be relatively prime, $\mathrm{GCD}(m_i, m_j) = 1$ if $i \neq j$.
(c) They must kept small.

EXAMPLE 2. For $p = 43$, $g = 3$, $m_0 = 2$, $m_1 = 3$, and $m_2 = 7$, determine the quotient of (36/2). This is a DRZ problem.

$\mathrm{ind}_3(36) = (0, 2, 0)$,	using Table 1 in ROM1 at ADD = 36,
$\mathrm{ind}_3(2) = (1, 0, 6)$,	using Table 1 in ROM2 at ADD = 2,
$\mathrm{ind}_3(36/2) = (0, 2, 0) - (1, 0, 6) = (1, 2, 1)$,	using mod2, mod3, and mod7 subtractors.

The results of the three moduli subtractors are converted to address for ROM3 as follows:

$$\mathrm{ADD}(1,2,1) = (1\ 10\ 001)_2 = 49,$$

using 1-bit, 2-bit, and 3-bit to represent mod2, mod3, and mod7 results. That is, at address ADD = 49 of ROM3, one obtains 18 in Table 2. The implementation of the DRZ process is illustrated in Figure 1. The size of each ROM is $2^6 \times 6$.

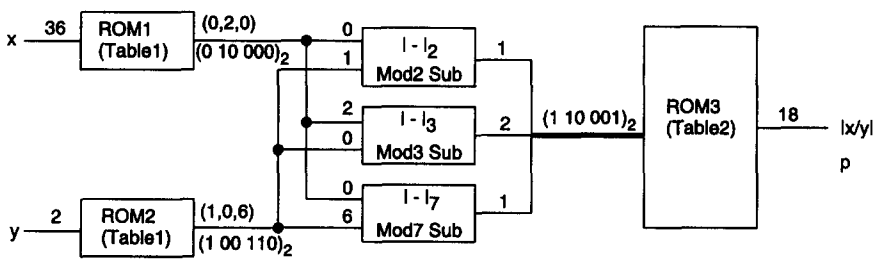

Figure 1. DRZ division in multimoduli RNS over $GF(p)$.

3.2. Scaling over Galois Field $GF(p)$

If the remainder is not zero, but it is known that the divisor is a product of some moduli, the division problem becomes a scaling problem. Division in any integer number system is defined by

$$x = \left\lfloor \frac{x}{y} \right\rfloor y + |x|_y, \tag{13}$$

where $\lfloor x/y \rfloor$ is the quotient, and $|x|_y$ is the remainder.

The quotient of scaling problem can be obtained as shown in the following theorem.

THEOREM 4. *In the scaling division problem, the divisor* $y = \prod_{i=0}^{k} m_i$ *is a product some moduli. The quotient* $\lfloor x/y \rfloor$ *can be computed as a sequence of DRZ problems with respect to each modulus in the product terms of* y,

$$\begin{aligned} q_0 &= \left\lfloor \frac{x}{m_0} \right\rfloor = \frac{x - |x|_{m_0}}{m_0}, \\ q_1 &= \left\lfloor \frac{q_0}{m_1} \right\rfloor = \left\lfloor \frac{x}{m_0 m_1} \right\rfloor = \frac{q_0 - |q_0|_{m_1}}{m_1}, \\ &\vdots \\ q_k &= \left\lfloor \frac{q_{k-1}}{m_k} \right\rfloor = \left\lfloor \frac{x}{m_0 m_1 \dots m_k} \right\rfloor = \frac{q_{k-1} - |q_{k-1}|_{m_k}}{m_k}, \end{aligned} \tag{14}$$

where $m_0 < m_1 < \cdots < m_k$.

PROOF. Starting with modulus m_0, x can be expressed in term of m_0,

$$x = \left\lfloor \frac{x}{m_0} \right\rfloor m_0 + |x|_{m_0} = q_0 m_0 + |x|_{m_0}. \tag{15}$$

Solving for q_0,

$$q_0 = \left\lfloor \frac{x}{m_0} \right\rfloor = \frac{x - |x|_{m_0}}{m_0}. \tag{16}$$

That is, q_0 is a DRZ problem. Dividing equation (16) by m_0,

$$\frac{x}{m_0} = q_0 + \frac{|x|_{m_0}}{m_0}. \tag{17}$$

Dividing equation (18) by m_1,

$$\frac{x}{m_0 m_1} = \frac{q_0}{m_1} + \frac{|x|_{m_0}}{m_0 m_1}, \qquad \left\lfloor \frac{x}{m_0 m_1} \right\rfloor = \left\lfloor \frac{q_0}{m_1} \right\rfloor, \tag{18}$$

since the second term is less than 1, q_0 can be expressed in term of m_1,

$$q_0 = \left\lfloor \frac{q_0}{m_1} \right\rfloor m_1 + |q_0|_{m_1} = q_1 m_1 + |q_0|_{m_1}.$$

Solving for q_1 and substituting equation (19),

$$q_1 = \left\lfloor \frac{q_0}{m_1} \right\rfloor = \frac{q_0 - |q_0|_{m_1}}{m_1} = \left\lfloor \frac{x}{m_0 m_1} \right\rfloor.$$

Again, this is a DRZ problem. Continuing in similar fashion q_{k-1} can be expressed in term of m_k,

$$q_{k-1} = \left\lfloor \frac{q_{k-1}}{m_k} \right\rfloor m_k + |q_{k-1}|_{m_k} = q_k m_k + |q_{k-1}|_{m_k}.$$

Solving for q_k,

$$q = q_k = \left\lfloor \frac{q_{k-1}}{m_k} \right\rfloor = \frac{q_{k-1} - |q_{k-1}|_{m_k}}{m_k} = \left\lfloor \frac{x}{m_0 m_1 \dots m_k} \right\rfloor = \left\lfloor \frac{x}{y} \right\rfloor.$$

That is, the last quotient of the DRZ sequence is the desired quotient of the scaling division problem. The remainder, rem, can be computed from the last quotient q,

$$\text{rem} = x - qy.$$

The general representation of y in terms of the r system moduli can be achieved by introducing the code CD defined as follows:

$$y = \prod_{i=0}^{r-1} (m_i)^{\mathrm{CD}(i)},$$

where

$$\mathrm{CD}(i) = \begin{cases} 1, & \text{if } m_i, \text{ is a product term of } y, \\ 0, & \text{otherwise.} \end{cases}$$

If $\mathrm{CD}(j) = 0$ and $\mathrm{CD}(j-1) = 1$, then

$$q_{j-1} = \left\lfloor \frac{q_{j-1}}{(m_j)^0} \right\rfloor (m_j)^0 + |q_{j-1}|_{(m_j)^0} = (q_j)(1) + 0 = q_j.$$

That is, the quotient with $\mathrm{CD}(j) = 0$ is the same as the last quotient with $\mathrm{CD}(k) = 1$ $(k < j)$, $q_j = q_k$. The hardware implementation for three moduli system is shown in Figure 2.

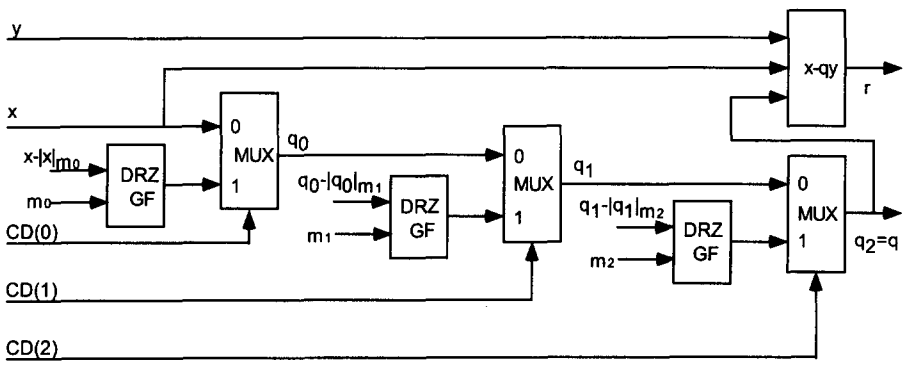

Figure 2. Scaling division problem over Galois field $GF(p)$.

NOTE. In the derivation of the Theorem 4, the order of DRZ sequence is not relevant, it is only required that the entire sequence be taken to obtain the final quotient q. For implementation purposes, we select the increasing order of moduli.

EXAMPLE 3. For $p = 43$, $g = 3$, $m_0 = 2$, $m_1 = 3$, and $m_2 = 7$, determine integer quotient value of (37/6). This is a scaling division problem, since

$$y = \prod_{i=0}^{2} (m_i)^{\mathrm{CD}(i)} = m_0 m_1 = (2)(3) = 6,$$

where $\mathrm{CD}(0) = 1$, $\mathrm{CD}(1) = 1$, and $\mathrm{CD}(2) = 0$,

$$q_0 = \left\lfloor \frac{x}{m_0} \right\rfloor = \frac{x - |x|_{m_0}}{m_0} = \frac{37 - |37|_2}{2} = \frac{36}{2} = 18 \text{ by DRZ } GF(43),$$

$$q_1 = \left\lfloor \frac{q_0}{m_1} \right\rfloor = \frac{q_0 - |q_0|_{m_1}}{m_1} = \frac{18 - |18|_3}{3} = \frac{18}{3} = 6 \text{ by DRZ } GF(43),$$

$q = q_2 = q_1 = 6$.

NOTE. With CD = 110, the first two stages in Figure 2 perform the DRZ computation by selecting input 1 of each MUX, and the last stage simply pass the previous quotient by selecting input 0 of its MUX.

The remainder rem is computed as follows:

$$\mathrm{rem} = x - qy = 37 - 6(6) = 1.$$

3.3. General Arithmetic Division over Galois Field $GF(p)$

General division is used when it is not known *a priori* that the dividend is a multiple of the divisor (DRZ) or when the divisor is a product of some moduli (scaling problem).

The general division problem x/y is first converted to scaling problem x/y', where y' satisfies the following conditions:

(1) $y' > y$,

(2) y' is a product of some moduli.

Table 3 shows that for moduli $\{2, 3, 7\}$ there are only $2^r - 1 = 2^3 - 1 = 7$ possible values of y', where r is the number of moduli. With a small number of entries, searching for y' for a given y is relatively simple and fast operation. For example, for divisor $y = 5$, Table 3 is searched for the first occurrence of $y' > y$. This occurs in ROM entry 2, where $y' = 6 = m_0 m_1 = (2)(3)$ with code CD = 110. The word size of the code ROM = size(y') + 3, where 3 is the 3-bit needed to store the code CD. In our example, the word size is 6 + 3 = 9, and address size is 3-bit to represent seven entries. That is, the ROM size used to implement Table 3 is $2^3 \times 9$.

Table 3. Look-up table for CDs for moduli {2,3,7}.

ADD	y'	CD m_0 2	m_1 3	m_2 7
0	2	1	0	0
1	3	0	1	0
2	6	1	1	0
3	7	0	0	1
4	14	1	0	1
5	21	0	1	1
6	42	1	1	1

THEOREM 5. *The maximum ratio between two consecutive y' is*

$$\left(\frac{y'(i)}{y'(i-1)}\right)_{\max} \le m_0 \qquad \text{and} \qquad \left(\frac{y'}{y}\right)_{\max} \le m_0.$$

PROOF. There are three cases to consider, from examining Table 3.

CASE 1. $m0 : 0 \Rightarrow 1$ is allowed, if $m1$ and $m2$ maintain their respective value.

$$\frac{y'(i)}{y'(i-1)} = \frac{(m_0)^1 (m_1)^{x1} (m_2)^{x2}}{(m_0)^0 (m_1)^{x1} (m_2)^{x2}} = m_0.$$

CASE 2. $m1 : 0 \Rightarrow 1$ is allowed, if $m0 : 1 \Rightarrow 0$ and $m2$ maintains its value.

$$\left(\frac{y'(i)}{y'(i-1)}\right) = \frac{(m_0)^0 (m_1)^1 (m_2)^{x2}}{(m_0)^1 (m_1)^0 (m_2)^{x2}} = \frac{m_1}{m_0}.$$

CASE 3. $m2 : 0 \Rightarrow 1$ is allowed, if $m0 : 1 \Rightarrow 0$ and $m1 : 1 \Rightarrow 0$.

$$\left(\frac{y'(i)}{y'(i-1)}\right) = \frac{(m_0)^0 (m_1)^0 (m_2)^1}{(m_0)^1 (m_1)^1 (m_2)^0} = \frac{m_2}{m_0 m_1}.$$

Therefore,

$$\left(\frac{y'(i)}{y'(i-1)}\right)_{\max} = \max\left\{m_0, \frac{m_1}{m_0}, \frac{m_2}{m_0 m_1}\right\} = m_0,$$

$$\left(\frac{y'(i)}{y}\right)_{\max} = \left(\frac{y'}{y}\right)_{\max} \le \left(\frac{y'(i)}{y'(i-1)}\right)_{\max} = m_0, \qquad \text{since } y'(i-1) < y.$$

Using the modified divisor y', the problem becomes a scaling problem. This procedure is repeated until at iteration k, the remainder, rem, is zero or quotient q is zero. The final quotient Q is the accumulation of each quotient q at each iteration plus the correction given in Theorem 6. ∎

THEOREM 6. *If the algorithm stops and* rem $> y$, *the correction of* Q *is given by*

$$Q = Q + (k - 1),$$

where k *is the smallest integer in the range* $1 < k \leq m_0$ *that satisfy* rem $< ky$.

PROOF. Correction is needed when $q = 0$ and rem $> y$. That is,

$$q = \left\lfloor \frac{\text{rem}}{y'} \right\rfloor = 0, \qquad \text{implies rem} < y',$$

since rem $> y$, then $y <$ rem $< y'$. Dividing by y yields

$$1 < \frac{\text{rem}}{y} < k \leq \frac{y'}{y} = m_0.$$

If k is the smallest integer that satisfy the inequality, then the quotient need to be corrected by adding

$$\left\lfloor \frac{\text{rem}}{y} \right\rfloor = k - 1.$$

That is, $Q = Q + (k - 1)$. ∎

LEMMA 3. *In the case of* $m_0 = 2$, *when the iteration stops, the final quotient* Q *is corrected as follows:*

$$Q = \begin{cases} Q, & \text{if rem} < y, \\ Q + 1, & \text{otherwise.} \end{cases}$$

3.4. General Division Algorithm Statement

The proposed division algorithm takes two unsigned integers x and y as inputs and returns the quotient $Q = [x/y]$, and the remainder $R = x \bmod y$. The algorithm GENDIV is outlined below.

Algorithm GENDIV

```
Input: (x, y)
Output: (Q, R)
Begin
A:      Convert to Scaling Problem: find y' from Table 3
B:      Initialization: x1 = x; Q = 0, R = 0
C:      SCALE (x1, y, CD, rem, q)
        Q = Q + q
        While (q ≠ 0) AND (rem ≠ 0) Do
        Begin
            rem = x1 − qy
            SCALE (x1, y, CD, rem, q)
            Q = Q + q
        End
D:      CORRECT(Q, y, rem)
        R = x − Qy
End.
```

EXAMPLE 4. For the moduli set $\{m_0, m_1, m_2\} = \{2, 3, 7\}$, $|37/5|_{43}$ can be computed as follows.

(A) Convert to Scaling Problem

$$x = 37,\ y = 5 \Rightarrow y' = 6,\ \text{CD} = (1, 1, 0) \text{ from Table 3.}$$

(B) Initialization: $x1 = 37$, $Q = 0$, $R = 0$.

(C) **Iteration 1.** Scaling problem 37/6

$$\begin{aligned}
x1 &= x = 37, \\
\text{CD}(0) &= 1,\ m0 = 2,\ q0 = 37 - |37|_2 = 36,\ q0 = 36/2 = 18, && \text{using DRZ over } GF(43), \\
\text{CD}(1) &= 1,\ m1 = 3,\ q1 = 18 - |18|_3 = 18,\ q1 = 18/3 = 6, && \text{using DRZ over } GF(43), \\
\text{CD}(2) &= 0,\ m2 = 7,\ q = q2 = q1 = 6, \\
Q &= Q + q = 0 + 6 = 6, && \text{accumulated quotient,} \\
\text{rem} &= x1 - qy = 37 - (6)(5) = 7.
\end{aligned}$$

Iteration 2. Scaling problem 7/6

$$\begin{aligned}
x_1 &= \text{rem} = 7, \\
\text{CD}(0) &= 1,\ q_0 = 7 - |7|_2 = 6,\ q_0 = 6/2 = 3, && \text{using DRZ over } GF(43), \\
\text{CD}(1) &= 1,\ q_1 = 3 - |3|_3 = 3,\ q_1 = 3/3 = 1, && \text{using DRZ over } GF(43), \\
\text{CD}(2) &= 0,\ q = q2 = q1 = 1, \\
Q &= Q + q = 6 + 1 = 7, \\
\text{rem} &= x_1 - qy = 7 - (1)(5) = 2.
\end{aligned}$$

Iteration 3. Scaling problem 2/6

$$\begin{aligned}
x1 &= \text{rem} = 2, \\
\text{CD}(0) &= 1,\ q_0 = 2 - |2|_2 = 2,\ q_0 = 2/2 = 1, && \text{using DRZ over } GF(43), \\
\text{CD}(1) &= 1,\ q_1 = 1 - |1|_3 = 0,\ q_1 = 0/3 = 0, && \text{using DRZ over } GF(43), \\
\text{CD}(2) &= 0,\ q = q2 = q1 = 0, \\
Q &= Q + q = 7 + 0 = 7.
\end{aligned}$$

Iteration stop since $q = 0$.

(D) Quotient Correction

$$\begin{aligned}
Q &= Q, \text{ since } \text{rem} = 2 < y = 5, \\
R &= x - Qy = 37 - (7)(5) = 2.
\end{aligned}$$

4. HARDWARE IMPLEMENTATION OF GENERAL DIVISION OVER $GF(p)$

The hardware implementation of the general division is shown in Figure 3. Its operation is described next. The hardware is started be sending the Start signal, a high going pulse, the dividend x is selected at input 1 of the MUX to the input $x1$ of the Scaling GF and the accumulator Q is cleared. The scaling computation proceeds until the remainder rem and quotient q are obtained. The Done signal is raised if rem $= 0$ or $q = 0$. If Done $= 1$, the process stops, otherwise it continues to the next iteration. In the next iteration, the remainder rem is feedback to input 0 of MUX and to the input $x1$ of Scaling GF, since in subsequent iteration the Start signal is zero. When the process stop (Done $= 1$), the value of accumulated quotient Q is corrected (Q is incremented by one if rem $> y$) and the final remainder ($R = x - Qy$) calculated. Finally, Q and R are latched.

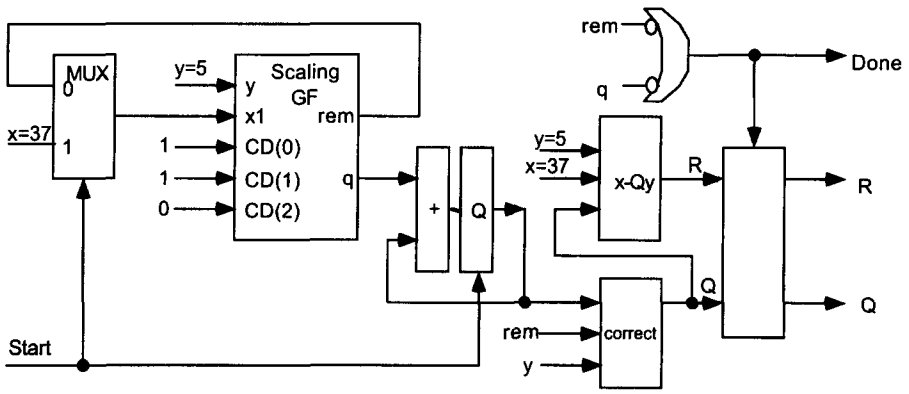

Figure 3. General division implementation for moduli {2,3,7}.

5. CONCLUSION

The proposed algorithm over Galois field $GF(p)$ provides an efficient algorithm for the general division problem. Efficient procedures were proposed to convert general division problem to scaling problem. The proposed algorithm and conversion procedure can be implemented by look-up tables, which means that division in RNS can be computed very fast. The results of this research work can be used to design a general purpose multimoduli ALU.

REFERENCES

1. N.S. Szabó and R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, McGraw-Hill, New York, (1967).
2. C. Hung and B. Parhami, Fast RNS division algorithms for fixed divisors with application to RSA encryption, *Information Processing Letters* **51**, 163–169 (1994).
3. C. Hung and B. Parhami, An approximate sign detection method for residue numbers and its application to RNS division, *Computers Math. Applic.* **27** (4), 23–35 (1994).
4. G.A. Julien, Residue number scaling and other operations using ROM arrays, *IEEE Trans. Comput.* **C-27** (4), 325–336 (1978).
5. E.D. Claudio, F. Piazza and G. Orlandi, Fast combinatorial RNS processors for DSP applications, *IEEE Trans. Comput.* **44** (5), 624–633 (1995).
6. S. Waser and M.J. Flynn, *Introduction to Arithmetic for Digital System Designers*, Holt, Rinehart & Winston, New York, (1982).
7. F.J. Taylor, Residue arithmetic: A tutorial with examples, *IEEE Comput.* **17** (5), 50–62 (1984).
8. M. Lu and J. Chiang, A novel division algorithm for the residue number system, *IEEE Trans. Comput.* **41** (8), 1026–1032 (1992).
9. W.A. Chren, A new residue number system division algorithm, *Computers Math. Applic.* **19** (7), 13–29 (1990).
10. D.K. Banerji, T.Y. Cheung and V. Ganesan, A high-speed division method in residue arithmetic, *IEEE Symp. Comput. Arithmetic.* (5), 158–164 (1981).
11. M.A. Hits and E. Kaltofen, Integer division in residue number systems, *IEEE Trans. Comput.* **44** (8), 983–989 (1995).
12. E. Kinoshita, H. Kosako and Y. Kojima, General division in the symmetric residue number system, *IEEE Trans. Comput.* **C-22**, 134–142 (1973).
13. M.L. Lin, E. Leiss and B. McInnis, Division and sign detection algorithm for residue number systems, *Computers Math. Applic.* **10** (4/5), 331–342 (1984).
14. D. Radhakrishnan and Y. Yuan, Novel approaches to the design of VLSI RNS multipliers, *IEEE Trans. Circuits and System—II* **39** (1), 52–57 (1992).
15. W. Beyer, *CRC-Standard Mathematical Tables and Formulae*, Edition 91–101, CRC Press, (1991).