Abstract: Until now many algorithms for division operation in residue number systems are presented but almost all of them have an overall loop, conversion from RNS to binary or mixed radix system or using LUT and exclude some numbers in the range of acceptable inputs as a denominator in division operation. In this paper, a non-iterative algorithm for division in RNS system is presented in which all numbers as denominator are accepted. Comparison based on more time consuming operation , modular multiplication , between proposed algorithm in this paper and [12] shows more than 2 times increase in speed and efficiency during divide operation.

Index Terms— residue number system; division; multiplicative inverse; Euclidean algorithm; base extension;

1. INTRODUCTION

Residue number system is used to present natural numbers using their remainders. In this system, some operations, such as addition and multiplication, are performed in parallel on remainders of inputs. Division, sign detection and number comparison are difficult operations in residue number systems. This weakness has limited the RNS to add, subtract and multiply operations. As a result, RNS is not used in many applications that need a divide operation. [1-4] Many algorithms for division in RNS are presented. Some of these iterative algorithms work by subtracting denominator from numerator in a major loop, until numerator gets less than denominator. Quotient is equal to the number of iterations of this major loop. Some of them use Newton iteration to compute reciprocal and then compute quotient. [5,6]

Another common way for division is using the definition of division. In this algorithm, first the position of the most-significant non-zero bit in the divisor and dividend is determined. Then, according to difference between these two positions, divisor is shifted to the left and is subtracted from dividend. These actions are repeated in a major loop until the result is smaller than divisor.

In some methods for dividing X to Y, first the proper \(2^k\) is detected such that \(Y.2^k \leq X \leq Y.2^{k+1}\). In the next iterations, these two margins varied until quotient obtained. [7-10]

2. A NON-ITERATIVE AND PURE RNS DIVISION ALGORITHM

During previous section many methods for division operation in RNS system introduced. These methods have some problems such as complexity, time consuming and nested loops for inputs. In this section a new method is introduced that solves these problems.

Consider \(X=(x_1,x_2,\ldots,x_L)\) and \(Y=(y_1,y_2,\ldots,y_L)\) as inputs of the algorithm. Modulo set is \(\{m_1,m_2,\ldots,m_L\}\) and output of algorithm is \(K=(k_1,k_2,k_3,\ldots,k_L)\) as quotient and \(R=(r_1,r_2,\ldots)\) as remainder.

Algorithm 1: the proposed pure RNS division algorithm

Input: \(X=(x_1,x_2,\ldots,x_L)\) , \(Y=(y_1,y_2,\ldots,y_L)\) and modulo set \(\{m_1,m_2,\ldots,m_L\}\)

Output: \(K=X/Y\) and \(R=(X \mod Y)\) in RNS

Condition: all moduli \(\{m_1,m_2,\ldots,m_L\}\) are prime numbers but 2.

Begin
1. Calculate \(Y^{-1} = \text{multiplicative inverse of} \ Y\)
2. Calculate \(R = X \mod Y\)
3. Calculate \(K = (X - R) \cdot Y^{-1}\)
4. If any \(y_i = 0\), then set \(k_i = (k_n \mod m_i)\) in which \(k_n \neq 0\).
End.

In Section 2.1 overall view of the proposed algorithm is presented. Modified multiplicative inverse algorithm is presented in Section 2.2. Section 2.3 discussed about calculating \(|X|_Y\) in pertinent details. In Section 2.4 special problem of having ‘zero’ in \(y_i\) is solved.
2.1 Overall views

Consider X and Y are two positive numbers. If X>Y and if X is divisible to Y, then K=X/Y=X Y \(^{-1}\). Variable Y \(^{-1}\) is the multiplicative inverse of Y, that is Y Y \(^{-1}\) = 1. It must be noticed that equation X/Y=X Y \(^{-1}\) is true when X is divisible to Y. Equation (1) through (4) proof this.

\[
X = Y.K + R \tag{1}
\]

Where K is natural number as quotient and R is the remainder of X/Y operation.

\[
X.Y^{-1} = K + R.Y^{-1} \tag{2}
\]

According to Eq.(2), if R=0 then:

\[
K = X.Y = X.Y^{-1} \tag{3}
\]

When X is not divisible to Y then, we must subtract R Y \(^{-1}\) from X Y \(^{-1}\) which is shown in equation (4).

\[
K = (X-R).Y^{-1} \tag{4}
\]

Example 1: Consider division of 20/4 in a RNS system which its modulo set is \{7, 17\}. Inputs to this RNS system are X=20=(6, 3) and Y=4=(4, 4). In order to compute X/Y, first multiplicative inverse of Y is calculated.

\[
Y^{-1} = 4^{-1} = (2, 13) \quad \text{where} \quad 4.4^{-1} = (1, 1)
\]

R=(20)\(_{4}\) = 0 = (0, 0)

Because R=0, K is calculated according to equation (3).

\[
K = X/Y = X.Y^{-1} = (6, 3) \quad (2, 13) = (5, 5) \quad \text{◆}
\]

2.2 Multiplicative inverse algorithm

Multiplicative inverse of a number Y which is denoted by Y \(^{-1}\) can be calculated using Euclidian algorithm which is used in [9] and is shown in algorithm (2).

Suppose input is in range \{0, M\} where M is the product of modulo set \{m\(_{1}\), m\(_{2}\), m\(_{3}\),..,m\(_{L}\)\}. Modular multiplication of Y \(^{-1}\) and Y is according to equation (5).

\[
Y.Y^{-1} = M.c + 1 \quad \tag{5}
\]

If one of the m\(_{i}\) is even, then M is even and M.c + 1 is odd. If Y is even there is not any Y \(^{-1}\) to multiply it to Y to have an odd result. In this situation there are some Y's without Y \(^{-1}\). As a result, even moduli eliminate numbers to have a multiplicative inverse.

If all the moduli are odd, M is odd. In this system the previous problem is solved, however there are some numbers without multiplicative inverse. The following example clarifies this situation.

Example 2: consider an RNS system with modulo set of \{11, 9\} multiplicative inverse of Y=21=(10, 3) is 21 \(^{-1}\) = (10, u). The letter 'u' denotes an undefined component in RNS presentation of a number. In this case 'u' indicates that for 3 in modulo 9 the multiplicative inverse is undefined because m\(_{i}\) =9 is multiple of y=3. ◆

To solve these problems the RNS system has to have modulus of prime numbers. In this condition, all numbers in the range of RNS have multiplicative inverse.

Lemma1: In a RNS system with modulo set of \{m\(_{1}\), m\(_{2}\), m\(_{3}\),..,m\(_{L}\)\}, all numbers in the range of \{0, M\} in which M= m\(_{1}\).m\(_{2}\).m\(_{3}\)....m\(_{L}\) has multiplicative inverse if and only if all moduli m\(_{i}\) are prime numbers but 2.

Proof: According to above discussion and examples, if all m\(_{i}\) are prime number but 2, then M is an odd number. Since all m\(_{i}\) are prime number, there is not any y\(_{i}\) in m\(_{i}\)=A.y\(_{i}\) where A is an integer number. Therefore, all numbers have multiplicative inverse.

If there is a m\(_{i}\) which is not prime number, or 2, then there is a y\(_{i}\) for m\(_{i}\)=A.y\(_{i}\) and y\(_{i}\) \(^{-1}\)=u. ◆

Based on lemma 1, the multiplicative inverse algorithm can be used for all numbers in the RNS system with prime moduli but 2.

Algorithm 2: the proposed modified Euclidean based multiplicative inverse algorithm

Input: y\(_{i}\), m\(_{i}\)
Output: inv\(_{i}\)
Condition: no condition
Begin

\[
g_0 = m_i, \quad g_1 = y_i \mod m_i, \quad v_0 = 0, \quad v_1 = 1;
\]

Loop

If g\(_{i}\) ≠ 0
then h = g\(_{0}\) div g\(_{1}\), g\(_{2}\) = g\(_{0}\) mod g\(_{1}\),
else inv\(_{i}\) = v\(_{0}\), output inv\(_{i}\) and halt;
\[
v_2 = h \times v_1;
\]
\[
v_0 = v_0 + m_i \times ((v_2 - v_0)/m_i);
\]
\[
v_2 = v_0 - v_2, \quad g_0 = g_1, \quad g_1 = g_2, \quad v_0 = v_1, \quad v_1 = v_2;
\]
End Loop

End

2.3 X mod Y

In order to calculate \(|X|\_Y\), base extension method is used. Suppose the primary modulo set is \{m\(_{1}\), m\(_{2}\), m\(_{3}\)\} and X=(x\(_{1}\), x\(_{2}\), x\(_{3}\)). Based on the Chinese remainder theorem, we have:

\[
X + m_1.m_2.m_3.d \mod m_4 = |x_1, m_1, m_2, m_3|^{x_1} + |x_2, m_1, m_2, m_3|^{x_2} + |x_3, m_1, m_2, m_3|^{x_3}
\]

Secondary modulo set is \{m\(_{1}\), m\(_{2}\), m\(_{3}\), m\(_{4}\)\} in which m\(_{4}\)=Y. In this new RNS system X=(x\(_{1}\), x\(_{2}\), x\(_{3}\), x\(_{4}\)) and x\(_{4}\)=|X|\_Y\) is unknown, which can be calculated using Eq. 7.

\[
x_4 = |x_1, m_1, m_2, m_3|^{x_1} + |x_2, m_1, m_2, m_3|^{x_2} + |x_3, m_1, m_2, m_3|^{x_3} + |x_4, m_1, m_2, m_3|^{x_4}
\]

Example 3: in a RNS with modulo set equal to \{3, 7, 13\}, X=23 is presented as X=(2, 2, 10). In order to calculate (X mod 5)=|X|\_5 it is not necessary to convert to binary system. The base extension technique can be used to find (X mod 5) as stated in equation (7):

\[
|3^{-1}|_3 = 2; \quad |7^{-1}|_3 = 3; \quad |13^{-1}|_3 = 2
\]
\[
|3^{-1} \times 7^{-1} \times 13^{-1} \times |X|_{13} + 4|_3 = 0
\]
\[
[2 \times 3 \times 2 \times |X|_{13}]_5 = 1
\]
\[ |X|_3 = 3 \]

Therefore for modulo set \{3, 5, 7, 13\}, \( X = (2, 3, 2, 10) \).

It must be noticed that if selected \( Y \) is small enough, calculating Eq.7 will be easy but if it is not small, Eq.7 can be simplified. This simplification depends on specific application which used a special structure in binary mode for \( Y \).

2.4 Calculating quotient

According to equation (4), quotient is \( K = (X - R)Y^{-1} \).

Example 4: calculate \( X/Y = 93/5 \) in RNS system in which \( M = \{7, 17\} \). In this system we have:

\[ Y = 5 = (5, 5), \quad Y^{-1} = (3, 7), \quad R = (X)Y = (93)5 = 3 \]

According to algorithm (2), we have \( X - R = (2, 8) - (3, 3) = (6, 5) \).

Since \( K = (X - R).Y^{-1} = (6, 5), (3, 7) = (4, 1) \), then quotient is equal to \( (4, 1) \) which in decimal system is 18.

Problem occurs when our number for calculating multiplicative inverse is multiple of one of the moduli. For example if \( Y = 17 \) and moduli are \( \{7, 17\} \), then \( Y^{-1} = (3, 0) \) and \( Y^{-1} = (5, 2) \). It means that the inverse of 0 is undefined, and it is denoted by ‘z’. This undefined value propagates through the algorithm and make the corresponding \( k_i = 0 \), which is not correct. In this situation \( k_i \) can be calculated using other nonzero and defined \( k_0 \) as:

\[ k_i = k_0 \mod m_i \]

Example 5 shows this situation.

Example 5: Consider \( X/Y = 80/7 \) in RNS system with modulo set of \( \{7, 17\} \). In this system, \( 7 = (0, 7), \quad 7^{-1} = (z, 5) \) and \( 80 = (80)_7 = 3 \).

According to algorithm (1), we have:

\[ (3, 12) - (3, 3) = (0, 9) \]

Then, \( K = 75/7 = (5, 2) \), and \( 0 = (z, 11) \).

Since \( k_1 = z \), we set \( k_1 = (k_2 \mod m_1) \). That is \( k_1 = (11)_{7} = 4 \)

Therefore, \( K = 80/7 = (4, 11) \).

Lemma 2: In division \( X \) to \( Y \) in a RNS system, \( K = X/Y \), if \( Y \) is multiple of one of the moduli, for example \( m_0 \), then \( y_i = 0 \) and \( k_i \) is undefined. In this situation, the proposed algorithm is executed for all moduli but \( m_0 \). In last step \( k_i \) is calculated using other non-zero \( k_0 \) as:

\[ k_i = k_0 \mod m_i \]

Proof: Consider that the selected moduli are \( \{m_1, m_2\} \) and \( m_2 > m_1 \). The greatest input of this system is \( N = m_2 - 1 \).

\[ X < m_2.m_2 \]

\[ Y = n.m_1 \]

\[ K = X/Y < (m_2.m_2/m_1) = m_2 \]

\[ k_i = |K|_{m_i} = K \]

\[ k_i = |K|_{m_i} = |k_j|_{m_i} \]

Example 6: consider dividing \( X/Y = 19/5 \) in a RNS system with modulo set of \( \{3, 7\} \). In this system, \( 5 = (2, 5), \quad 5^{-1} = (2, 3) \) and \( 19 = (19)_{7} = 4 \).

According to algorithm (1), we have:

\[ (1, 5) - (1, 4) = (0, 1) \]

Then, \( Q = 19/5 = (0, 1), \quad (2, 3) = (0, 3) = 3 \)

3. ANALYSES AND COMPARISON:

In this section we compare the presented algorithm in this paper with algorithm presented in [12]. In [12] there is an iterative algorithm to calculate the division of \( X \) to \( Y \). All numbers in the range of RNS are not acceptable as denominator, \( Y \). However, in the presented non-iterative algorithm, all numbers in the range of system can be set as a denominator.

The analysis is based on the number of modular multiplication, because modular multiplication is the most time consuming operations in this conversion. We suppose that this algorithm is implemented on the parallel processors.

According to the algorithm (1), first \((R = X \mod Y) \) and \( (Y^{-1} = \text{inverse of } Y) \) are to be computed. According to [12] for computing \( R = (X \mod Y) \), \( m+1 \) modular multiplications and for \( Y^{-1} = \text{inverse of } Y \), \( d \) modular multiplications are needed. Because these two operations are independent, we can compute them in parallel. Therefore, according to our proposed algorithm, maximum \((m+1, d) \) time delay is needed. The last modular multiplication is for \( K = (X - R).Y^{-1} \).

The complexity of our proposed algorithm is shown in table 1.

Suppose the overall loop in the division algorithm of [12] is repeated \( n \) times. Table 2 shows the complexity analysis of this algorithm.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time of modular multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R = X \mod Y )</td>
<td>( \max(m+1, d) )</td>
</tr>
<tr>
<td>( K = (X - R).Y^{-1} )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>Total time</td>
<td>( \max(m+2, d+1) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time of modular multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \rightarrow a' ); ( b \rightarrow b' )</td>
<td>( m+1 )</td>
</tr>
<tr>
<td>( b' \rightarrow (b')^{-1} )</td>
<td>( d )</td>
</tr>
<tr>
<td>( c' \rightarrow c ); ( (b')^{-1} \rightarrow (b'')^{-1} )</td>
<td>( m+1 )</td>
</tr>
<tr>
<td>((a (a'')^{-1} - c)(p^{-1}) ); ( (b (b'')^{-1})(p^{-1}) )</td>
<td>( 2 )</td>
</tr>
<tr>
<td>Total time</td>
<td>( (2m+d+4)n )</td>
</tr>
</tbody>
</table>

The overall delay time of the algorithm presented in paper [12] is \((2m+d+4)n \) where our proposed algorithm has the overall delay time of \( \max (m+2, d+1) \). Suppose \( d+1 \) is greater than \( m+2 \). The ratio of these delays, called \( s \), is:

\[ s = \frac{(2m+d+4)n}{(d+1)} \]

\[ (8) \]
Figure 5 shows this delay ratio versus number of bits, $d$, which is varied from 2 to 32. In this figure, number of major loop iterations, $n$, is considered to be 2 and figures are for different number of moduli of the RNS system, $m$. According to figure 5, proposed algorithm in this paper has more than 2 times increase in speed and efficiency to paper [12] during divide operation.

4. CONCLUSION

Many division algorithms which proposed for residue number systems, have three problems: an overall loop and not supporting all numbers in the range as a denominator. In this paper, we proposed a non-iterative and pure RNS division algorithm which solves these problems. Comparison between presented algorithm and [12], showed more than 2 times increase in speed and efficiency during divide operation.

REFERENCES