# A High-Speed Division Algorithm For Residue Number System

Ahmad A. Hiasat and Hoda S. Abdel-Aty-Zohdy, PhD.
Microelectronics System Design Laboratory
Department of Electrical and Systems Engineering
Oakland University, Rochester, MI 48309-4401

**Abstract:** A new algorithm for one of the long-standing problems in Residue Number System, namely division, is presented. The algorithm is so simple. It approaches the paper-and-pencil division procedure where the quotient is selected to guarantee a non-negative remainder. This algorithm does NOT require sign and overflow detection , scaling, or redundant moduli. Based on computer simulation results, the algorithm is four times faster than the most recent and competitive published work [8].
**Keywords:** Residue Number System, Division, Fractional Representation, Hardware Implementation.

## I. INTRODUCTION

Residue Number System (RNS) has the advantage of carry-free arithmetic operations. Thus, using residue arithmetic, in principle, would increase the computer processing speed. In particular, addition, subtraction and multiplication can be performed on each residue digit concurrently and independently. However, there are drawbacks associated with RNS. These drawbacks include the difficulty of arithmetic operations like division, sign and overflow detection. Generally speaking, all reported algorithms[1]-[8] have the disadvantage of lengthy arithmetic operations, large execution time and/or hardware requirements. The complexity of these algorithms arises due to utlizing Mixed-Radix Conversion (MRC) technique, and performing difficult residue operations. In this paper, the necessity for MRC, scaling, and sign/overflow detection is eliminated, with reduction in the number of iterations. Hence, the division in RNS has been significantly simplified, and therefore, the execution time and the hardware requirements will, eventually, be less.

## II. ALGORITHM

Assume that X, Y and Q are non-negative integers such that $Q = \lfloor \frac{X}{Y} \rfloor, Y \neq 0$ , then the following steps introduce the basic idea for division in RNS (define $h(I)$ to be the highest power of 2 in the variable I):

1. Set Quotient Q to zero; Q = 0.
2. Find $k$ , where $k = h(Y)$ expressed in $r$ bits, $r = \lceil log_2(log_2 M) \rceil$.
3. Find $j$ , where $j = h(X)$ expressed in $r$ bits.
4. If $j > k$, then:
   $Q' = Q + 2^{j-k-1}$. $X' = X - 2^{j-k-1} * Y$.
   $Q = Q'$, $X = X'$
   Go To Step 3.
5. If $j = k$, then:
   $X' = X - Y$, $j' = h(X')$ , so if $j' < j$ then $Q = Q + 1$. Otherwise, Q is unchanged. End procedure.
6. If $j < k$, then Q is unchanged. End procedure.

This basic algorithm can be used efficiently in RNS division arithmetic. One of the important feature of the algorithm is the selection of the quotient to be $2^{j-k-1}$, to guarantee a non-negative remainder. Unlike all other algorithms this, in principle, will eliminate all hardware and execution time needed for sign determination and overflow detection operations.

### A. Realization of the Algorithm in RNS

The realization of this new algorithm is still based upon converting the residue representation of the dividend and the divisor to a weighted code in order to derive some information regarding the highest power of 2 contained in a residue number. The fractional representation technique of the Chinese Remainder Theorem (CRT) is less expensive than MRC to derive this basic information. This technique was introduced by Soderstrand [9] and Van Vu [10]. The scheme is suitable for large dynamic ranges which can NOT be decoded using a single table. To have this paper self-contained, the technique will be introduced very briefly [9]-[10].

**Fractional Representation Technique:** The CRT can be written as [4]:

$$X = \sum_{i=1}^{N} \frac{M}{m_i} \left| \frac{x_i}{\hat{m}_i} \right|_{m_i} - Mp \qquad (1)$$

where:$\{m_1, m_2, \ldots, m_N\}$ , moduli set of N relatively prime positive integers. $M = \prod_{i=1}^{N} m_i$, dynamic range. $X \overset{RNS}{\rightarrow} (x_1, x_2, \ldots\ldots, x_N)$. $x_i = |X|_{m_i}$, least positive remainder when dividing X by $m_i$. $\hat{m}_i = \frac{M}{m_i}$. $|\frac{1}{\hat{m}_i}|_{m_i}$, multiplicative inverse of $m_i$ (i.e. $|\frac{m_i}{\hat{m}_i}|_{m_i} = 1$). $p$ is a non-negative integer. Dividing both sides of eq.(1) by $M$ for unsigned conversion, yields:

$$\frac{X}{M} = \sum_{i=1}^{N} \frac{1}{m_i} \left| \frac{x_i}{\hat{m}_i} \right|_{m_i} - p \qquad (2)$$

Unsigned integers are considered in this paper, however, the same concepts can still be applied for signed integers. Hence, the value of $\frac{X}{M}$ can be obtained by evaluating the right hand side of eq.(2). This is basically done by letting each $x_i$ addresses a table which stores $\frac{1}{m_i}|\frac{x_i}{\hat{m}_i}|_{m_i}$. The output of these N tables can be added using a fast multioperand adder[11]. Any integer overflow resulting from this adder is disregarded since it represents the value p (multiplies of M). The fractional value stored in tables should be expressed using t bits where t $\geq \lceil log_2 MN \rceil$ if M is odd and t $\geq \lceil log_2 MN \rceil - 1$ otherwise [10].

The realization can be described as follows:

1. Apply the residue digits of the divisor Y to the fractional representation circuit to obtain $Y/M$. This requires a memory cycle followed by a multioperand addition (see Fig.(1)).

2. Obtain k using a proper combinational circuit like a priority encoder (not shown in Fig.(1)).

3. Apply the residue digits of the dividend (remainder) to the fractional representation circuit to obtain $X/M$. A memory access cycle and a multioperand addition is also needed here. Obtain j, (we will denote $h(0) = 0$).

4. Apply j and k ( a total of $2r$ bits), or their difference, to a RAM or PAL that will produce $Q_i$. $Q_i$ is the residue representation of $X_e/Y_e$, where: $X_e$ is an estimated value of X given by: $X_e = 2^j M$. Similarly, $Y_e$ is an estimated value of Y given by: $Y_e = 2^{1+k} M$. Or equivalently $Q_i = 2^{j-k-1}$.

5. The output of the table, $Q_i$, is applied to a residue multiplier to get $Q_i Y$.

6. The output of the multiplier $(Q_i Y)$ is subtracted from $X$ using a residue subtractor.

7. The output of the residue subtractor is applied again to the fractional representation circuit (i.e. step 3 above) as long as $j > k$.

8. Whenever $j = k$, the output of the RAM (PAL) is 1. The new remainder would be $X' = X - Y$. The residue adder will or will not be incremented depending on the following: the value of $X'$ is applied to the fractional representation circuit to produce $j'$, where $j'$ is the highest power of 2 contained in $X'$. If $j' = -1$ , then this indicates that an overflow has taken place, and the residue adder should not be incremented. However, if $j' \neq -1$, then the residue adder should be incremented by 1. After either case, the iterative procedure is stopped. The output of the adder is $Q$.

9. It is obvious that for the case $j < k$, then $X < Y$.

**Example:** For the moduli set $\{11,14,15\}$, $M = 2310$. Divide $X = (1212)$ by $Y = (33)$. $Y \overset{RNS}{\rightarrow} (0, 5, 3)$ and $X \overset{RNS}{\rightarrow} (2, 8, 12)$

- **1st iteration**

  1. $Y = (0, 5, 3)_{RNS} \overset{Y/M}{\rightarrow} (.000000111010)_2$, $k = h(Y/M) = -7$.
  2. $X = (2, 8, 12)_{RNS} \overset{X/M}{\rightarrow} (.101\cdots\cdots)_2$, $j = h(X/M) = -1$.
  3. Thus $Q_1 = 2^{-1+7-1} = 32 \overset{RNS}{\rightarrow} (10, 4, 2)$. $Q = (0, 0, 0) + (10, 4, 2) = (10, 4, 2)$. (i.e. $Q = 32$)
  4. $X = X - Q_1 Y = (2, 2, 6)$, ( i.e. $X = 156$).

- **2nd iteration**

  1. $X = (2, 2, 6)_{RNS} \overset{X/M}{\rightarrow} (.000100\cdots\cdots)_2$, $j = h(X/M) = -4$.
  2. Thus $Q_2 = 2^{-4+7-1} = 4 \overset{RNS}{\rightarrow} (4, 4, 4)$. $Q = (10, 4, 2) + (4, 4, 4) = (3, 8, 6)$. (i.e. $Q = 36$)
  3. $X = X - Q_2 Y = (2, 10, 9)$ ( i.e. $X = 24$)

- **3rd iteration**

  1. $X = (2, 10, 9)_{RNS} \overset{X/M}{\rightarrow} (.0000001\cdots)_2$, $j = h(X/M) = -7$.

2. Since $j = k$, then $X' = X - Y = (2,5,6)$, (i.e $X' = M + 24 - 33 = 2301$) and $j' = -1$, thus Q is not changed. $Q = (3,8,6)$. ( i.e. $Q = 36$).

## B. Proof Of Correctness Of The Algorithm

Before introducing the proof of the algorithm, the following theorem has also to be proved:

**Theorem 1** *In RNS, for any fractional representations $\frac{X}{M}, \frac{Y}{M}$ where $X, Y \in [0, M)$, $j = h(\frac{X}{M})$, $k = h(\frac{Y}{M})$ and $j = k$ then $X \geq Y$ if $j' \neq -1$, and $X < Y$ if $j' = -1$, where $j' = h(\frac{X-Y}{M})$.*

**proof:**

Since $\frac{X}{M}$ and $\frac{Y}{M}$ are of the same order (i.e j=k) that is:

$2^j \leq \frac{X}{M} < 2^{j+1}$ , and similarly $2^j \leq \frac{Y}{M} < 2^{j+1}$

**For the case** $X \geq Y$ : $|X - Y|_M = X - Y \geq 0$, then:

$0 \leq \frac{X-Y}{M} < 2^j$. Hence, $j' = h(\frac{X-Y}{M}) < j$ , or equivalently; $j' < j$.

Since the highest value of j is -1, then: $j' \leq -2$

Note that for the special case, $\frac{X-Y}{M} = 0$, we will denote $j'$ to be 0.

Consequently, if $X \geq Y$ then $j' \neq -1$.

**For the case** $X < Y$ :

$|X - Y|_M = M - (Y - X)$, then $j' = h(\frac{M+X-Y}{M})$. or $j' = h(1 - \frac{Y-X}{M})$.

but since $X < Y$, then: $0 < \frac{Y-X}{M} < 2^j$, or $j' \geq h(1 - 2^j)$.

Since $X, Y \leq M$, then the maximum value of $j$ is -1. Therefore, $0 > j' \geq h(\frac{1}{2})$. Or: $j' = -1$.

**The proof of the algorithm is the following:**

- For the case $j > k$: The estimated value of X given by $X_e = 2^j M$ is guaranteed to be less or equal to the actual dividend (remainder) X. Similarly: the estimated value of Y given by $Y_e = 2^{k+1} M$ is guaranteed to be greater or equal to the actual divisor Y. Thus, $\frac{X}{Y} \geq \lfloor \frac{X_e}{Y_e} \rfloor$. In other words, the remainder is guaranteed to be non-negative.
- For the case j=k, two possibilities are expected:
  1. $X < Y$. This case is detected according to the above theorem by evaluating $j'$. Hence, if $j' = -1$ then the quotient should not be incremented. Procedure is then stopped.
  2. $X \geq Y$. If $j' \neq -1$ then the quotient is incremented by 1. Procedure is then stopped.
- For the case $j < k$, it is obvious that $X < Y$. Procedure is stopped.

## III. EVALUATION

In order to evaluate the performance of the new algorithm, it has to be compared with other RNS division algorithms. Chren's algorithm [3] has a slightly better performance compared to another competitive one [2]. Nevertheless, it requires MRC and residue scaling operations for each iteration. If look-up tables are used, then MRC and scaling would require (N-1) and (2N-1) memory cycles respectively [12]. Chren's algorithm requires also a redundant modulus, which represents another drawback.

Gamberger[5] presented an algorithm which does not use MRC. The number of iterations for each division problem is proportional to the magnitude of the divisor. Hence, the mean of the number of iterations is very high. Moreover, the hardware implementation suggested by Gamberger is very complicated and expensive. Lu's algorithm [8] is the most recent work. It does NOT use MRC. However, it utilizes the idea of fractional representation of $X/M$ to detect the parity of a residue number, and hence to check if an overflow has taken place. It is the most comparative algorithm to compare our algorithm with.

Following the method of measuring execution time for residue arithmetic algorithms [2-4], comparison is based on calculating the mean of the basic residue arithmetic operations needed by any competitive algorithm. These basic operations are: residue addition, subtraction and multiplication. Following the same prevalent method [2-4], memory accesses are not counted. Lu's algorithm requires $2log_2 Q$ steps where Q is the quotient. Each step consists of several residue additions and subtractions, one residue multiplication, two memory access cycles and one or two multioperand additions. The new algorithm requires $log_2 Q$ steps where each step consists of one residue multiplication $(Q_i Y)$ , one residue subtraction $(X - Q_i Y)$ that is performed in parallel with one residue addition $(Q = Q + Q_i)$, one multioperand addition, and two memory access cycles: one to get the fractional representations of residue digits, while the other is to obtain $Q_i$. To compare performances of the new algorithm and Lu'u algorithm, computer simulation programs have been developed to calculate the average number of Residue Operations (RO) for different moduli sets. The mean of Multioperand Additions (MA) has been calculated and compared. Two moduli sets were selected, $M1 = (11, 13, 15, 19, 23, 29, 31)$, and $M2 = (29, 31, 43, 47, 53, 55, 59, 61, 63)$. These moduli sets were selected to serve different dynamic ranges. A sample of 200 million randomly generated numbers

Table 1: Simulation Results

| Moduli | Mean of RO | | Mean of MA | |
|---|---|---|---|---|
| Set | Our's | Lu's | Our's | Lu's |
| M1 | 2.67 | 10.36 | 3.041 | 5.496 |
| M2 | 2.72 | 10.39 | 3.089 | 5.504 |

within the dynamic range defined by each set were simulated. The results of simulation are listed in Table 1.

Table 1 indicates, explicitly, that the new algorithm is four times faster regarding the number of basic residue arithmetic operations. Moreover, the average number of multioperand additions needed by this algorithm is almost half that needed by the other algorithm.

For moduli sets where all the bits of residue digits can be applied simultaneously to a single RAM (i.e. no multioperand additions are needed), this algorithm would still be four times faster. Furthermore, as densities and speed of RAMs increase, the use of larger look-up tables would be very advantageous. This algorithm makes the possibility of building RNS dividers feasible and beneficial. The improved memory density, will enhance the speed and expand the dynamic range of an implemented divider.

## IV. CONCLUSION

A new efficient algorithm for RNS division is presented. The algorithm does not require sign and overflow detection, scaling, or redundant moduli. Therefore, the complexity of division in RNS has been substantially reduced. Based on computer simulation results, this algorithm is much faster than the most recent one. Since there is no single implementation of a RNS divider has been published yet, the proposed hardware implementation makes the possibility of building RNS dividers very feasible and practical. This, in fact, would enable RNS to be a stronger force in the field of computer design.

## References

[1] E. Kinoshita, H. Kosako, and Y. Kojima, "General division in the symmetric residue number system,". *IEEE Trans. Comput., C-22, pp 134-142, Feb. 1973.*

[2] D. Banerji, T. Cheung, and V. Ganesan, "A high-speed division method in residue arithmetic,". *in Proc. 5th IEEE Symp. Comput. Arithmetic, 1981, pp 158-164.*

[3] W. Chren Jr., "A new residue number division algorithm,". *Computer Math. Appl., vol. 19, no. 7, pp 13-29, 1990.*

[4] N. Szabo, and R. Tanaka, *"Residue Arithmetic and Its Applications to Computer Technology,"*. McGraw Hill, New York, 1967.

[5] D. Gamberger, "New approach to integer division in residue number system,". *in Proc. of 10th Symp. on Comput. Arith., 1991, pp 84-91.*

[6] Y. Kier, P. Cheney and M. Tannenbaum, "Division and overflow detection in residue number systems,". *IRE Trans. Electron. Comput., EC-11, pp 501-507, Aug. 1962.*

[7] L. Lin, E. Leiss, and B. Mcinnis, "Division and sign detection algorithm for residue number systems,". *Comput. Math. Appl., 10, pp 331-342, 1984.*

[8] M. Lu, and J. Chiang, "A novel division algorithm for residue number system,". *IEEE Trans. Compu., C-41, pp 1026-1032, Aug. 1992.*

[9] M. Soderstrand, C. Vernia, and J. Chang, "An improved residue system digital-to-analog converter,". *IEEE Trans Circ. and Sys., CAS-30, pp 903-907, Dec. 1983.*

[10] T. Van Vu, "Efficient implementations of chinese remainder theorem for sign detection and residue decoding,". *IEEE Trans Compu., C-34, pp 646-651, July 1985.*

[11] D. Atkins, and S. Ong, "Time-component complexity of two approaches to multioperand binary addition,". *IEEE Trans. Compu., C-28, pp 918-926, Dec. 1979.*

[12] M. Soderstrand, M. A., W. Jenkins, G. Jullien, F. Taylor, Eds. *"Residue Number System Arithmetic: Modern Applications in Digital Signal Processing,"*. IEEE Press, New York, 1986.

[13] A. Hiasat, and H. Abdel-Aty-Zohdy, "Design and implementation of a fast and compact residue-based semi-custom VLSI arithmetic chip,". *Presented at the $37^{th}$ Midwest Symp. on Cir. and Sys., Aug. 3-5, 1994, Lafayette, LA.*

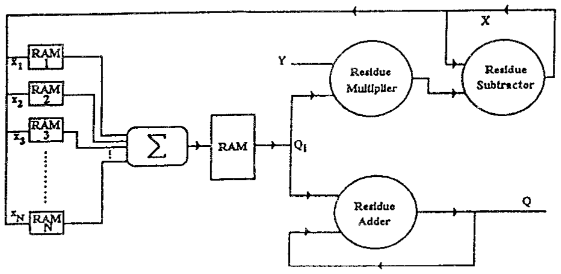

Fig.(1): Proposed Implementation of the algorithm