Generalized Matrix Method for Efficient Residue to Decimal Conversion

Kazeem Alagbe Gbolagade[1,2], Member, IEEE and Sorin Dan Cotofana[1], Senior Member IEEE,
1. Computer Engineering Laboratory,
Delft University of Technology,
The Netherlands.
2. University for Development Studies, Navrongo, Ghana.
E-mail: {gbolagade,sorin}@ce.et.tudelft.nl

***Abstract*—In this paper, we present a matrix based method for efficient Residue to Decimal Conversion. First, we generalize a previously proposed technique that was restricted to 5-moduli set such that it becomes applicable to any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$. Next, we simplify the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS functional units. For an n-digit RNS number $X = (x_1, x_2, x_3, ..., x_n)$ the proposed method takes at most n iterations. Each iteration requires one parallel subtractions and 2 multiplications except the first one. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(n^2\right)$. In particular, the utilization of our technique, for 3-moduli and 10-moduli RNS results in the reduction of the total number of arithmetic operations required by the conversion process with 13.33% and 66.05%, respectively, when compared to state of the art MRC.**

***Index Terms*—Residue Number System, Mixed Radix Conversion, Matrix Method, Arithmetic Operations.**

I. INTRODUCTION

There is performance degradation in computing hardware built based on Weighted Number Systems (WNSs) due to the carry propagation phenomenon inherent to WNSs. The reduction/elimination of carry chains is the major challenge in improving computer arithmetic performance. Several approaches have been proposed, e.g., carry lookahead, prefix calculations, anticipated calculation, and alternative number representation systems, e.g., (redundant) signed digit systems, or Residue Number Systems (RNS). RNS, which is the theme of this paper has interesting inherent characteristics such as parallelism, modularity, fault tolerance, and carry free operations and for this reason it has been utilized in Digital Signal Processing (DSP) applications such as digital filtering, convolutions, correlation, fast Fourier transforms, and image processing [1], [11]. RNS processors however have to make use of data conversion, which has to be implemented as fast as possible in order not to nullify the RNS advantages.

The work on residue to binary conversion is based on Chinese Remainder Theorem (CRT) [2]-[8], [10] or on Mixed Radix Conversion (MRC) [9], [12], [14]. CRT is desirable because the computation can be parallelized while MRC is by its very nature a sequential process. However many up to date RNS to binary/decimal converters are based on MRC due to the complex and slow modulo-M operation (M being the system dynamic range thus a rather large constant) required by CRT. The main problem with the MRC is that the computations of the MR digits is done in a serial manner and requires a large number of arithmetic operations.

In this paper, we present a matrix based method for efficient Residue to Decimal Conversion. First, we generalize a previously proposed technique that was restricted to 5-moduli set such that it becomes applicable to any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$. Next, we simplify the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS functional units. For an n-digit RNS number $X = (x_1, x_2, x_3, ..., x_n)$ the proposed method takes at most n iterations. Each iteration, except the first one, requires 2 multiplications and one parallel subtraction over all the mod-m_i ways of the RNS adder. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(n^2\right)$. In particular, the utilization of our technique, for 3-moduli and 10-moduli RNS results in the reduction of the total number of arithmetic operations required by the conversion process with 13.33% and 66.05%, respectively, when compared to state of the art MRC. These results provide a wide range of applications of RNS in the design of DSP intensive applications.

The rest of the paper is organised as follows: We briefly present the necessary background in Section II. Section III describes the proposed Generalized Matrix Method for efficient residue to decimal conversion. We evaluate the performance of our proposal in Section IV while the paper is concluded in Section V.

II. BACKGROUND

RNS is defined in terms of a set of relatively prime moduli set $\{m_i\}_{i=1,n}$ such that $\gcd(m_i, m_j) = 1$ for $i \neq j$, where gcd means the greatest common divisor of m_i and m_j, while $M = \prod_{i=1}^{n} m_i$, is the dynamic range. The residues of a decimal number X can be obtained as $x_i = |X|_{m_i}$ thus X can be represented in RNS as $X = (x_1, x_2, x_3..., x_n)$, $0 \leq x_i < m_i$. This representation is unique for any integer $X \in [0, M-1]$.

978-1-4244-2342-2/08/$25.00 ©2008 IEEE.

We note here that in this paper we use $|X|_{m_i}$ to denote the X mod m_i operation and the operator Θ to represent the operation of addition, subtraction, or multiplication. Given any two integer numbers K and L RNS represented by $K = (k_1, k_2, k_3, ..., k_n)$ and $L = (l_1, l_2, l_3, ..., l_n)$, respectively, $W = K\Theta L$, can be calculated as $W = (w_1, w_2, w_3, ..., w_n)$, where $w_i = |k_i \Theta l_i|_{m_i}$, for $i = 1, n$. This means that the complexity of the calculation of the Θ operation is determined by the number of bits required to represent the residues and not by the one required to represent the input operands. This creates the premises for high speed arithmetic and for example it has been proved that RNS based addition can be performed in $O(\log(\log(n)))$ delay for unrestricted moduli and in $O(\log(n))$ for 2^n and $2^n - 1$ moduli [13].

Conversion from RNS to decimal is relatively fast using MRC as it does not involve the large modulo-M calculations present in CRT. MRC is formulated as follows [1]: Suppose that we have a set of residues$\{x_1, x_2, x_3, ...x_n\}$ with the corresponding set of moduli $\{m_1, m_2, m_3, ...m_n\}$ and a set of digits $\{a_1, a_2, a_3, ..., a_n\}$ which are the Mixed Radix Digits (MRD), the decimal equivalent of the residues can be computed as follows:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 + ... + a_n m_1 m_2 m_3 ... m_{n-1} \tag{1}$$

where the mixed radix digits are given as follows:

$$\begin{aligned} a_1 &= x_1 \\ a_2 &= \left| (x_2 - a_1) \left| m_1^{-1} \right|_{m_2} \right|_{m_2} \\ a_3 &= \left| \left((x_3 - a_1) \left| m_1^{-1} \right|_{m_2} - a_2 \right) \left| m_2^{-1} \right|_{m_3} \right|_{m_3} \end{aligned} \tag{2}$$

.

.

$a_n = |((...(x_n - a_1)|m_1^{-1}|_{m_2} - a_2)|m_2^{-1}|_{m_3} - ...$
$-a_{n-1})|m_{n-1}^{-1}|_{m_n}|_{m_n}$

For the MRD a_i, $0 \leq a_i < m_i$, any positive number in the interval [0, $\Pi_{i=1}^{N} m_i - 1$] can be uniquely represented. It can be inferred from the description above and in line with [12] that a total of $\frac{n(n-1)}{2}$ arithmetic subtractions and multiplications are required in order to compute the MRD for n-moduli sets.

The matrix method described in [9] for the moduli set $\{11, 7, 5, 3, 2\}$, can be also utilized for conversion. It operates on a number of jumps $J_i, i = 1, 5$ and it was introduced in [9] only by means of an example, the RNS number $(4, 0, 1, 2, 1)$, as follows:

$J_1 = 4$, the first residue and the first location is given by:

$$X - 4 = \begin{pmatrix} |4-4|_{11} = 0 \\ |0-4|_7 = 3 \\ |1-4|_5 = 2 \\ |2-4|_3 = 1 \\ |1-4|_2 = 1 \end{pmatrix}$$

The second jump is defined by a number J_2 such that:

$J_2 = k_2.11$ and $|3 - J_2|_7 = 0$, giving $k_2 = 6$.

The second location is therefore given by:

$$X - 4 - 66 = \begin{pmatrix} |0-66|_{11} = 0 \\ |3-66|_7 = 0 \\ |2-66|_5 = 1 \\ |1-66|_3 = 1 \\ |1-66|_2 = 1 \end{pmatrix}$$

The procedure is repeated until all the elements in the final location are zeros, i.e.,

$$X - 4 - 66 - 231 - 385 - 1155 = \begin{pmatrix} |0|_{11} = 0 \\ |0|_7 = 0 \\ |0|_5 = 0 \\ |0|_3 = 0 \\ |0|_2 = 0 \end{pmatrix}$$

From this description, one can observe that 4 modular equations must be solved in order to obtain J_is. The required decimal number is given by summing up the J_is. In the next section we present a generalized matrix method which is applicable to any moduli set. Furthermore, we simplify the process of finding the jumps J_i and we show that this process is similar in every stage.

III. PROPOSED MATRIX METHOD (MATR)

Our approach is based on the periodicity property inherent in RNS. Periodicity is the ability of numbers to cycle in fixed periods with respect to some given moduli and within the dynamic range of the system. For example, given a 3-moduli set $\{m_1 = 5, m_2 = 4, m_3 = 3\}$, the residues cycle in a basic period. That is, the residue sequence of modulus m_1 will have a period of 5 entries (0,1,2,3,4), m_2 will have a period of 4 entries (0,1,2,3) while m_3 will have a period of 3 entries (0,1,2). Based on that the decimal equivalent of any given residue number is obtained by jumping backwards in the residue table (a table containing all the possible decimal numbers together with their residue equivalent within the dynamic range of the system) to the nearest residue number with at least one residue being zero. The value of that jump is recorded and the process continues until all the residues become zeros. Suppose that we have the residue set $\{x_1, x_2, x_3, ..., x_n\}$ corresponding to the moduli set $\{m_1, m_2, m_3, ..., m_n\}$, we can calculate its decimal counterpart by a maximum number of n-consecutive jumps in the residue table such that each jump increases the number of zero residues with one. The process continues until in the final location all the elements become zero.

More formally, given the moduli set $\{m_1, m_2, m_3, ..., m_n\}$, the residue number $(x_1, x_2, x_3, ..., x_n)$ is converted into the decimal number X as follows:

$$X = \sum_{i=1}^{n} p_i, \tag{3}$$

where p_i is defined as

$$\begin{aligned} p_i &= (m_1 m_2 ... m_{i-1}) \left| (m_1 m_2 ... m_{i-1})^{-1} \right|_{m_i} \\ &\quad \left| t_{(i-1)j} \right|_{m_i}, \end{aligned} \tag{4}$$

$i > 1$ and $t_{(i-1)j}$ is a value to be determined based on the the following matrix based computation.

If the jumps are $p_1, p_2, p_3, ..., p_n$ and they correspond to the residues $x_1, x_2, x_3, ..., x_n$, respectively, then,

$p_1 = x_1$ and the first location is:

$$X - p_1 = \begin{pmatrix} |x_1 - p_1|_{m_1} = 0 \\ |x_2 - p_1|_{m_2} = t_1 \\ |x_3 - p_1|_{m_3} = t_2 \\ |x_4 - p_1|_{m_4} = t_3 \\ |x_5 - p_1|_{m_5} = t_4 \\ . \\ . \\ |x_n - p_1|_{m_n} = t_{n-1} \end{pmatrix} \tag{5}$$

The second jump and is given by $p_2 = c_2 m_1$, where c_2 has to satisfy $|t_1 - p_2|_{m_2} = 0$. Solving the above two equations we obtain that

$$p_2 = m_1 \left| \left| (m_1)^{-1} \right|_{m_2} t_1 \right|_{m_2} \tag{6}$$

and the second location is defined by:

$$X - p_1 - p_2 = \begin{pmatrix} |0 - p_2|_{m_1} = 0 \\ |t_1 - p_2|_{m_2} = 0 \\ |t_2 - p_2|_{m_3} = t_{21} \\ |t_3 - p_2|_{m_4} = t_{31} \\ |t_4 - p_2|_{m_5} = t_{41} \\ . \\ . \\ |t_n - p_2|_{m_n} = t_{(n-1)1} \end{pmatrix} \tag{7}$$

The third jump is determined as $p_3 = c_3 m_1 m_2$ and $|t_{21} - p_3|_{m_3} = 0$ and it is given by:

$$p_3 = (m_1 m_2) \left| \left| (m_1 m_2)^{-1} \right|_{m_3} t_{21} \right|_{m_3} \tag{8}$$

The third location is therefore given by:

$$X - p_1 - p_2 - p_3 = \begin{pmatrix} |0 - p_3|_{m_1} = 0 \\ |0 - p_3|_{m_2} = 0 \\ |t_{21} - p_3|_{m_3} = 0 \\ |t_{31} - p_3|_{m_4} = t_{311} \\ |t_{41} - p_3|_{m_5} = t_{411} \\ . \\ . \\ \left|t_{(n-1)1} - p_3\right|_{m_n} = t_{(n-1)11} \end{pmatrix} \tag{9}$$

This iterative process continues until the final location is zero and the last jump is determined by $p_n = c_n m_1 m_2 m_3 ... m_{n-1}$ and $\left|t_{(n-1)11...} - p_n\right|_{m_n} = 0$.

Solving the above, p_n, which is needed in the last matrix computation is given by:

$$p_n = (m_1 m_2 ... m_{n-1}) \left| \left| (m_1 m_2 ... m_{n-1})^{-1} \right|_{m_n} t_{(n-1)11} \right|_{m_n} \tag{10}$$

Then the final location is given by:

$$X - p_1 - p_2 - p_3 ... - p_n = \begin{pmatrix} |0 - p_n|_{m_1} = 0 \\ |0 - p_n|_{m_2} = 0 \\ |0 - p_n|_{m_3} = 0 \\ |0 - p_n|_{m_4} = 0 \\ |0 - p_n|_{m_5} = 0 \\ . \\ . \\ \left|t_{(n-1)11...} - p_n\right|_{m_n} = 0 \end{pmatrix} \tag{11}$$

Therefore, as stated in Equation (3), the required Decimal equivalent of $(x_1, x_2, x_3, ...x_n)$ with respect to the moduli set $\{m_1, m_2, m_3, ...m_n\}$ is given by:

$$X = p_1 + p_2 + p_3 + ... + p_n \tag{12}$$

A critical look at Equations (6), (8), and (10) indicates that this process is similar, i.e., it is simply the product of moduli and their multiplicative invereses, which is precomputed together with a number $t_i, i = 2, n$. This is similar to the process of computing MRD but the subtractions are done in parallel.

In order to clarify the algorithm let us assume for example that we want to convert $(3, 2, 0)_{RNS(5|4|3)}$

to decimal. The algorithm is applied as follows:

(i) $(3, 2, 0)_{RNS(5|4|3)}$

$p_1 = 3$

$X - 3 = \begin{pmatrix} |3 - 3|_5 = 0 \\ |2 - 3|_4 = 3 \\ |0 - 3|_3 = 0 \end{pmatrix}$

p_2 is computed by:

$p_2 = 5 \, |3|_4 = 15$

The next location is therefore given by:

$X - 3 - 15 = \begin{pmatrix} |0 - 15|_5 = 0 \\ |3 - 15|_4 = 0 \\ |0 - 15|_3 = 0 \end{pmatrix}$

The final location is already (0,0,0) , there is no need to proceed further and hence the result is $X = 3 + 15 = 18$, as it should.

IV. PERFORMANCE EVALUATION

In the proposed technique presented in Section III, computation of p_1 is straight forward as $p_1 = x_1$. This implies that the computation of at most $n - 1$ p_is is required in the conversion process, as in some cases as in the previous example the conversion may need less steps. The computation of each of the $p_i, i = 2, n$ requires 2 multiplications because each $p_i, i = 2, n$ is simply the product of the moduli and their multiplicative invereses, which are pre-computed together with a number $t_i, i = 2, n$. This can be clearly seen from Equations (6), (8) and (10). In addition, the conversion process also requires n-parallel subtractions. Given that each of those subtractions is done modulo-m_i they can be executed in parallel on the RNS adder embedded in the RNS processor. The summation of $p_i's$ also requires $(n - 1)$ additions. This implies that the total number of computations that are required sums up to at most $4n - 3$. Hence, the asymptotic complexity of the proposed technique is in the order of $O(n)$. The total number of operations is computed based on the assumption that an addition takes one cycle and a multiplication two cycles

Method	Moduli	Reduction [in %]
MATR	3	13.33
MATR	4	29.63
MATR	5	40.48
MATR	6	48.33
MATR	7	54.32
MATR	8	59.05
MATR	9	62.88
MATR	10	66.05

Table I
ARITHMETIC OPERATIONS REDUCTION IN %

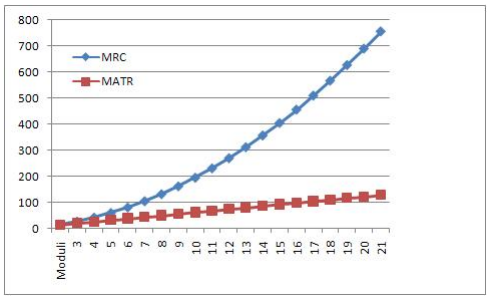

Figure 1. Number of Arithmetic Operation vs Moduli Set Length

thus we consider that one multiplication is equivalent delay wise with two additions.

For the MRC, $\frac{n(n-1)}{2}$ additions and $\frac{n(n-1)}{2}$ multiplications are required for the computation of MRD in addition to $n-1$ additions and $n-1$ multiplications required by Equation (1) to compute the required decimal number while for the MATR n subtractions, $2(n-1)$ multiplications are required in addition to the $n-1$ additions required by Equation (12) to compute the required decimal number.

Figure I shows the behaviour of both the MRC and MATR methods in terms of the number of arithmetic operations as the length of the moduli set increases. As the moduli set cardinality increases the number of arithmetic operation in MRC grows quadratically while for MATR it increases with a constant factor of six arithmetic operations.

In Table I, we present the percentage reduction of the total number of arithmetic operations required by MATR when compared to MRC. One can observe that MATR achieves 13.33% and 66.05% reductions with moduli set of length three and ten, respectively. As expected the larger the number of moduli in the RNS the larger the reduction the proposed conversion method exhibits.

We note here that in Table I, the following notations are utilized: Moduli - stands for the number of moduli in the considered RNS; Reduction - stands for reduction of the total number of arithmetic operations in percentage achieved by MATR over the traditional MRC.

V. CONCLUSIONS

In this paper, we presented a matrix based method for efficient Residue to Decimal Conversion. First, we generalized a previously proposed technique that was restricted to 5-moduli set such that it can be utilized in conjunction with any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$.

Next, we simplified the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS functional units. For an n-digit RNS number $X = (x_1, x_2, x_3, ..., x_n)$ the proposed method takes at most n iterations. Each iteration, except the first one, requires 2 multiplications and one parallel subtraction over all the mod-m_i ways of the RNS adder. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(n^2\right)$. In particular, the utilization of our technique, for 3-moduli and 10-moduli RNS results in the reduction of the total number of arithmetic operations required by the conversion process with 13.33% and 66.05%, respectively, when compared to state of the art MRC. Given that the method we proposed substantially reduces the RNS to binary/decimal conversion overhead it makes RNS more effective in addition and multiplication dominated DSP applications.

REFERENCES

[1] Szabo, N., and Tanaka, R. : Residue arithmetic and its application to computer technology, McGraw-Hill, New York, 1967.

[2] W. Wang, M.N.S. Swamy, M.O. Ahamad and Y. Wang: A high - Speed residue-to-binary converter and a Scheme for its VLSI Implementation. Proceedings of ISCAS, page330-333, 1999.

[3] A.A. Hiasat and H.S. Abdel-Aty-Zohdy: Residue-to-binary Arithmetic Converter for the Moduli Set (2k,2k-1,2k+1):IEEE Transactions on Circuits and Systems-II Analog and Digital Signal Processing, Vol.45, No. 2, Feb. 1998.

[4] W. Wang, M.N.S. Swamy, and M.O. Ahmad:An Area-Time efficient residue-to-binary converter. Proceedings of 43rd IEEE MidWest Symp. On Circuits and Systems, Lansing MI, Aug.8-11,2000.

[5] A.Skavantzos and Y Wang: New Efficient RNS-to-Weighted Decoders for Conjugate- Pair Moduli Residue Number Systems. Proceedings of ISCAS, pages 1345-1350, 1999.

[6] A.Skavantzos and Y Wang: Implementation Issues of the Two-Level Residue Number System with pair of Conjugate Moduli. IEEE Transactions on Signal Processing, Vol. 47, No.3, March 1999.

[7] M. Abdallah and A.Skavantzos : On the Binary Quadratic Residue System with Non-coprime Moduli. IEEE Transaction on Signal Processing, Vol. 45, No. 8, August 1997.

[8] A. Skavantzos and M Abdallah: Novel Residue Arithmetic Processors for High Speed Digital Signal Processing. Proceedings of ISCAS,pages 187-193, 1998 .

[9] H.M. Yassine: Fast Arithmetic based on Residue Number System Architectures. IEEE ISCAS '99, pages 2947-2950, Singapore,1999

[10] A.A. Hiasat: Efficient Residue to Binary Converter. IEEE Proceedings-Digital Tech, 2003.

[11] P.V. Ananda Mohan: Residue Number Systems Algorithms and Architectures, The Kluwer Int. Series in Eng. Ans Science, the Netherlands, 2002.

[12] H.M. Yassine and W.R. Moore: Improved mixed-radix conversion for residue number architectures, IEEE proceedings, Vol. 138, No.1 pp120-124, Feb. 1991.

[13] B. Parhami: Computer Arithmetics:Algorithms ans Hardware Designs, New York, Oxford University Press, 2000.

[14] H.M. Yassine: Matrix Mixed-Radix Conversion For RNS Arithmetic Architectures. 34th Midwest Symposium on Circuits and Systems, pages 273-278,1992.