International Journal of Computer Applications (0975 - 8887)
Volume 141 - No.5, May 2016

An Algorithm for Magnitude Comparison in RNS based on Mixed-Radix Conversion II

Konstantin Isupov
Department of Electronic Computing Machines, Vyatka State University
610000, Kirov, Russia

ABSTRACT

The residue number system (RNS) has computational advantages for large integer arithmetic because of its parallel carry free, and high-speed arithmetic nature. However, magnitude comparison is a very complex operation for RNS. This paper presents a new comparison algorithm based on the modification of Mixed-Radix Conversion II technique. The new algorithm uses small modulo operations only and has a linear time complexity in terms of the size of the moduli set.

General Terms

Computer arithmetic, algorithm design and analysis

Keywords

Residue number system, magnitude comparison, mixed-radix conversion, MRC-II

1. INTRODUCTION

The residue number system (RNS) [1, 2] as a non-positional alternative to binary representation is popular in many high performance arithmetic applications, such as digital signal processing [3], cryptography [4], multiple errors detection and correction [5]. RNS is defined by a set of moduli $\{m_1, m_2, \ldots, m_n\}$ that are relatively prime integers (i.e., the greatest common divisor between any two moduli is one). The dynamic range of an RNS is given by $M = \prod_{i=1}^{n} m_i$. Any integer X from 0 to $M-1$ is represented in RNS by the n-tuple $\langle x_1, x_2, \ldots, x_n \rangle$, where $x_i = |X|_{m_i}$ for all $i = 1, 2, \ldots, n$ and $|X|_{m_i}$ is defined as $x_i \equiv X \bmod m_i$. Addition, subtraction and multiplication on different moduli m_i are done in parallel:

$$Z = X \; op \; Y \rightarrow \begin{cases} z_1 = |x_1 \; op \; y_1|_{m_1}, \\ z_2 = |x_2 \; op \; y_2|_{m_2}, \\ \ldots \\ z_n = |x_n \; op \; y_n|_{m_n}. \end{cases}$$

As a consequence, operations on large wordlengths can be split into several modular operations executed in parallel and with reduced wordlength [6]. Organizing such a level of parallelism using the positional number systems is obviously not possible.

Unfortunately, due to the non-positional nature of the RNS, some arithmetic operations such as magnitude comparison, division and overflow detection are more complex in RNS than in conventional binary systems. This difficulty prevents a wide variety of general-purpose computations from taking advantage of the residue arithmetic [7]. In this paper a brief survey of the methods for comparing the magnitudes of numbers in RNS is presented, as well as a new parallel comparison algorithm, which is based on the modified Mixed-Radix Conversion II technique. The proposed algorithm uses a small modulo m_i operations only and has $O(n)$ time complexity, where n is the size of the moduli set.

2. RNS MAGNITUDE COMPARISON METHODS

The traditional technique for comparing numbers in RNS is based on the Chinese remainder theorem (CRT) and involves computing the binary representation of the number [2]:

$$X = \left| \sum_{i=1}^{n} w_i |\alpha_i x_i|_{m_i} \right|_M.$$

where $w_i = M/m_i$, and α_i is the multiplicative inverse of w_i with respect to m_i. In large dynamic ranges this method becomes slow as it requires costly long integer multiplications by constants, additions and large modulo M operation.

Mixed-Radix Conversion (MRC) is an alternative technique used to estimate the magnitude of RNS numbers [1]. MRC involves converting a number from RNS with moduli set $\{m_1, m_2, \ldots, m_n\}$ to a Mixed-Radix System (MRS) with bases $m_1, m_1 m_2, \ldots, m_1 m_2 \cdots m_{n-1}$, i.e., finding coefficients $a_1, a_2, \ldots, a_n$ that satisfy the expression:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 + \cdots + a_n m_1 m_2 \cdots m_{n-1}. \quad (1)$$

Since MRS is a positional number system, it is easy to compare the magnitudes of numbers in it. Assume that $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ are MRS representations of RNS numbers X and Y, respectively. In this case, $X > Y$ when $a_n > b_n$, and $X < Y$ when $a_n < b_n$; if $a_n = b_n$ then the digits a_{n-1} and b_{n-1}, etc. should be compared in a similar way. If $a_i = b_i$ for all $i = n, n-1, \ldots, 1$, then X and Y are equals. Thus, comparison of RNS numbers reduces to computing their MRS representations. However, the classical MRC is a slow sequential method, which requires $O(n^2)$ arithmetic operations. Therefore, with a large n, its application would require high computational costs.

Another approach is the Sum-of-Quotients Technique (SQT) [8]. This approach formulates the problem of estimation of the magnitudes of RNS numbers in terms of computing the indices of diagonals laying out the n-dimensional space formed by RNS moduli. The main drawback of SQT is that it requires handling of large $(\log_2 SQ)$-bit values, where $SQ = \sum_{i=1}^{n} M/m_i$.

Other techniques for number comparison in RNS are based on the parity checking [9] or the core functions [10]. The core functions require an iterative process of descent and lifting to find the critical core value. A different solution [9] to do the residue number comparison assumes that all moduli of the moduli set are odd and ROM lookup tables (LUTs) are mandatory to resolve the difficulty in the determination of the operand parity [7]. There are methods of number comparison for RNS with some special moduli sets [7, 11]. Nevertheless, the scope of such systems is limited, particularly when a large dynamic range is required.

3. MIXED-RADIX CONVERSION II

MRS representation for a number in n-moduli RNS can be computed using the classical Szabo-Tanaka MRC algorithm [1]. This algorithm requires a total of $n(n-1)/2$ residue arithmetic subtractions and also a total of $n(n-1)/2$ residue multiplications. However, these multiplications cannot be eliminated or reduced, and, moreover, these multiplications are nested operations, and so cannot be performed in parallel [12]. Therefore, Szabo-Tanaka MRC is considered inefficient for large n values.

In 2007, Akkal and Siy proposed a new technique for conversion of RNS numbers to binary system through MRS, called MRC-II [13]. According to MRC-II, the following recurrence is computed:

$$X_i = a_i M_{i-1} + X_{i-1}, \quad 2 \le i \le n,\ n \ge 2, \tag{2}$$

where a_i is the i-th MRC coefficient, $M_{i-1} = \prod_{k=1}^{i-1} m_k$ and $X_1 = a_1 = x_1$. Eventually X_n is a binary representation of X. All coefficients a_i are computed in advance and stored in LUTs. Each i-th LUT (Tab_{m_i}) is of the size $m_i \log_2 m_i$. The LUT index is the value $t_i = |x_i - X_{i-1}|_{m_i}$ for $i \ge 2$. The LUT contents are defined by the equality $|x_i - X_{i-1}|_{m_i} = |a_i M_{i-1}|_{m_i}$.

Algorithm 1 computes MRS representation of RNS number according to MRC-II.

Algorithm 1. Akkal-Siy Mixed-Radix Conversion

Input: $X = \langle x_1, x_2, \ldots, x_n \rangle$
Output: $(a_1, a_2, \ldots, a_n)$ satisfying the conditions of (1)
Precomputation: LUTs $Tab_{m_2}, \ldots, Tab_{m_n}$ that contain all values of a_i. $M_i = \prod_{k=1}^{i} m_k$ for all $1 \le i \le n-2$.

$a_1 \leftarrow X_1 \leftarrow x_1$
$t_2 \leftarrow |x_2 - X_1|_{m_2}$
$a_2 \leftarrow Tab_{m_2}(t_2)$
for $i = 3$ **to** n **do**
  $X_{i-1} \leftarrow a_{i-1} M_{i-2} + X_{i-2}$
  $t_i \leftarrow |x_i - X_{i-1}|_{m_i}$
  $a_i \leftarrow Tab_{m_i}(t_i)$
end for
return $(a_1, a_2, \ldots, a_n)$

Thus $O(n)$ operations only should be performed to compute all MRC coefficients for X. Unfortunately, the described algorithm is impractical since large integers M_{i-2} should be manipulated to determine the table index t_i, which is used to select the MRC coefficient a_i. Accordingly, X_{i-1} will also be large, making it hard and laborious to compute the difference between x_i and X_{i-1} followed by modulo m_i reduction. A modified algorithm without above drawback is discussed in the next Section.

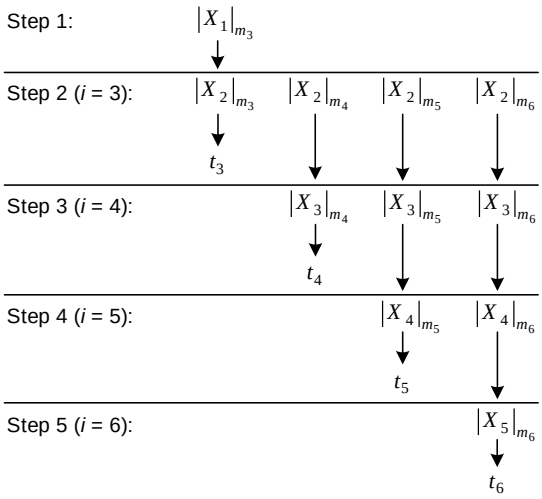

Fig. 1. Computation of the LUT indices t_i for six-moduli RNS.

4. PROPOSED ALGORITHM FOR COMPARING NUMBERS IN RNS

Since completely convert of the RNS numbers into binary form is not required for their comparison, to determine the index t_i it will only be enough to compute $|X_{i-1}|_{m_i}$ instead of X_{i-1}. From (2) it follows that:

$$|X_{i-1}|_{m_i} = \left|a_{i-1} |M_{i-2}|_{m_i} + |X_{i-2}|_{m_i}\right|_{m_i} \tag{3}$$

for $3 \le i \le n$ and $n \ge 3$.

Now, $t_i = \left|x_i - |X_{i-1}|_{m_i}\right|_{m_i}$ is an index for selecting a_i from the table Tab_{m_i}. Constants $|M_i|_{m_j} = \left|\prod_{k=1}^{i} m_k\right|_{m_j}$ for all $1 \le i \le n-2$ and $i+2 \le j \le n$ are computed in advance. Algorithm 2 computes the MRS representation of RNS number based on (3).

Algorithm 2. Modified Mixed-Radix Conversion

Input: $X = \langle x_1, x_2, \ldots, x_n \rangle$
Output: $(a_1, a_2, \ldots, a_n)$ satisfying the conditions of (1)
Precomputation: LUTs $Tab_{m_2}, \ldots, Tab_{m_n}$. Constants $|M_i|_{m_j}$ for all $1 \le i \le n-2$ and $i+2 \le j \le n$.

$a_1 \leftarrow X_1 \leftarrow x_1$
$t_2 \leftarrow |x_2 - X_1|_{m_2}$
$a_2 \leftarrow Tab_{m_2}(t_2)$
$|X_1|_{m_3} \leftarrow |x_1|_{m_3}$
for $i = 3$ **to** n **do**
  for $j = i$ **to** n **do in parallel**
    $|X_{i-1}|_{m_j} \leftarrow \left|a_{i-1} |M_{i-2}|_{m_j} + |X_{i-2}|_{m_j}\right|_{m_j}$
  end for
  $t_i \leftarrow |x_i - |X_{i-1}|_{m_i}|_{m_i}$
  $a_i \leftarrow Tab_{m_i}(t_i)$
end for
return $(a_1, a_2, \ldots, a_n)$

The $|X_{i-1}|_{m_j}$ variable, computed in the inner loop when $i = j$, is used to determine LUT index t_i, while other variables, computed in the inner loop when $i < j \leq n$, are used in the next iteration of the outer loop. For example, Fig. 1 shows the process of computing the LUT indices t_i ($i \geq 3$) for an RNS with six-moduli set.

All iterations of inner loop are mutually independent and are executed concurrently. All operations are carried out using modulo m_i operations only. So, Algorithm 2 allows to determine the MRS representation for a number within a n-moduli RNS in the order of $O(n)$ time.

Finally, the Algorithm 3 implements a comparison of numbers.

Algorithm 3. RNS Number Comparison

Input: $X = \langle x_1, x_2, \ldots, x_n \rangle, Y = \langle y_1, y_2, \ldots, y_n \rangle$
Output: $X > Y$ or $X < Y$ or $X = Y$
Precomputation: Same as those for Algorithm 2.
1: Compute $(a_1, a_2, \ldots, a_n)$ using Algorithm 2 {for X}
2: Compute $(b_1, b_2, \ldots, b_n)$ using Algorithm 2 {for Y}
3: **for** $i = n$ **downto** 1 **do**
4: **if** $a_i > b_i$ **then**
5: **return** $X > Y$
6: **else if** $a_i < b_i$ **then**
7: **return** $X < Y$
8: **end if**
9: **end for**
10: **return** $X = Y$

5. EXAMPLE

Let us consider the comparison of $X = 251 = \langle 6, 8, 9, 4 \rangle$ and $Y = 815 = \langle 3, 5, 1, 9 \rangle$ in RNS with moduli set $\{7, 9, 11, 13\}$. For these moduli $M_1 = 7, M_2 = 63, M_3 = 693$.

Precomputation stage:

Firstly, let us calculate Tab_{m_i} for moduli $9, 11, 13$:

—Tab_9 items are calculated according to the following condition: $|a_2 \cdot 7|_9 = t_2$, where $t_2 \in \{0, 1, 2, \ldots, 8\}$. Tab_9 items are calculated as follows:

if $t_2 = 0$ then $a_2 = 0$ since $|0 \cdot 7|_9 = 0$,
if $t_2 = 1$ then $a_2 = 4$ since $|4 \cdot 7|_9 = 1$,
if $t_2 = 2$ then $a_2 = 8$ since $|8 \cdot 7|_9 = 2$,
if $t_2 = 3$ then $a_2 = 3$ since $|3 \cdot 7|_9 = 3$,
if $t_2 = 4$ then $a_2 = 7$ since $|7 \cdot 7|_9 = 4$,
if $t_2 = 5$ then $a_2 = 2$ since $|2 \cdot 7|_9 = 5$,
if $t_2 = 6$ then $a_2 = 6$ since $|6 \cdot 7|_9 = 6$,
if $t_2 = 7$ then $a_2 = 1$ since $|1 \cdot 7|_9 = 7$,
if $t_2 = 8$ then $a_2 = 5$ since $|5 \cdot 7|_9 = 8$.

Thus, $Tab_9 = \{0, 4, 8, 3, 7, 2, 6, 1, 5\}$.

—Tab_{11} items are calculated according to the following condition: $|a_3 \cdot 63|_{11} = t_3$, where $t_3 \in \{0, 1, 2, \ldots, 10\}$. Thus, $Tab_{11} = \{0, 7, 3, 10, 6, 2, 9, 5, 1, 8, 4\}$.

—Tab_{13} items are calculated according to the following condition: $|a_4 \cdot 693|_{13} = t_4$, where $t_4 \in \{0, 1, 2, \ldots, 12\}$. Thus, $Tab_{13} = \{0, 10, 7, 4, 1, 11, 8, 5, 2, 12, 9, 6, 3\}$.

Now let us determine $|M_i|_{m_j}$ for $i = 1, 2$ and $j = i + 2, \ldots, 4$:

$$|M_1|_{11} = 7, \quad |M_1|_{13} = 7, \quad |M_2|_{13} = 11.$$

Table 1. Performance Evaluation of Different Residue Comparison Algorithms.

Size of moduli set (n)	Number of required modulo operations		
	Szabo-Tanaka	Yassine-Moore	Algorithm 3
4	28	20	18
8	120	76	46
12	276	164	74
16	496	284	102
20	780	436	130
24	1128	620	158
28	1540	836	186
32	2016	1084	214

Mixed-Radix Conversion and Comparison stages:

MRS representations of X and Y in accordance with the Algorithm 2 are computed as follows:

For $X = \langle 6, 8, 9, 4 \rangle$:	For $Y = \langle 3, 5, 1, 9 \rangle$:
$a_1 = X_1 = x_1 = \mathbf{6}$	$b_1 = Y_1 = y_1 = \mathbf{3}$
$t_2 = \|8 - 6\|_9 = 2$	$t_2 = \|5 - 3\|_9 = 2$
$a_2 = Tab_9(2) = \mathbf{8}$	$b_2 = Tab_9(2) = \mathbf{8}$
$\|X_2\|_{11} = \|8 \cdot 7 + 6\|_{11} = 7$	$\|Y_2\|_{11} = \|8 \cdot 7 + 3\|_{11} = 4$
$\|X_2\|_{13} = \|8 \cdot 7 + 6\|_{13} = 10$	$\|Y_2\|_{13} = \|8 \cdot 7 + 3\|_{13} = 7$
$t_3 = \|9 - 7\|_{11} = 2$	$t_3 = \|1 - 4\|_{11} = 8$
$a_3 = Tab_{11}(2) = \mathbf{3}$	$b_3 = Tab_{11}(8) = \mathbf{1}$
$\|X_3\|_{13} = \|3 \cdot 11 + 10\|_{13} = 4$	$\|Y_3\|_{13} = \|1 \cdot 11 + 7\|_{13} = 5$
$t_4 = \|4 - 4\|_{13} = 0$	$t_4 = \|9 - 5\|_{13} = 4$
$a_4 = Tab_{13}(0) = \mathbf{0}$	$b_4 = Tab_{13}(4) = \mathbf{1}$

Thus, $(6, 8, 3, 0)$ is the MRS representation of X and $(3, 8, 1, 1)$ is the MRS representation of Y. Since $a_4 < b_4$, it can be concluded that $X < Y$.

6. PERFORMANCE EVALUATION

The main difference between the existing algorithms for comparing the magnitudes of numbers in RNS and the represented one lies in the fact that modified parallel Mixed-Radix Conversion technique for computation of MRS representations (Algorithm 2) is used in the latter. As mentioned above, classical Szabo-Tanaka MRC algorithm requires $n(n-1)/2$ modulo arithmetic subtractions and the same number of modulo arithmetic multiplications, which are being performed strictly sequentially [1]. In 1991, Yassine and Moore proposed an improved MRC algorithm, which also requires $n(n-1)/2$ subtractions, but only $n-2$ multiplications [12]. Algorithm 2 requires only $n-2$ additions, $n-2$ multiplications and $n-1$ subtractions, provided that all iterations of inner loop will be performed simultaneously, divided into n parallel channels. Thus, both classical Szabo-Tanaka MRC and improved Yassine-Moore MRC have a time complexity of $O(n^2)$, where n is the size of the moduli set. In contrast, Algorithm 2 has a time complexity of $O(n)$.

Two MRS representations need to be computed to compare RNS numbers. Additionally, n operations will be required (in worst case scenario) for pairwise comparison of all the MRC coefficients. The total number of operations required for comparing numbers in RNS by the different algorithms is shown in Table 1. Fig 2 show the estimated speedup of the algorithm proposed in this paper.

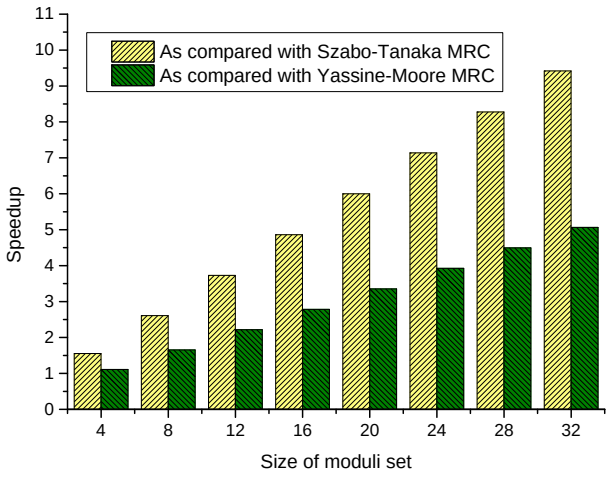

Fig. 2. Estimated speedup of the proposed algorithm.

It can be seen that if the hardware provides considerable room for parallelism (computations can be divided into n threads), the new algorithm is substantially faster than its counterparts.

Classical MRC-II approach (Algorithm 1) also has a time complexity of $O(n)$, but requires operating with large integers. The presented algorithm operates with small m_i moduli instead.

In 2009, Gbolagade and Cotofana proposed an MRC-algorithm, which is closest to Algorithm 2 [14]. It is also characterized by the linear time complexity in terms of the moduli set size, and is substantially a parallel implementation of Szabo-Tanaka MRC. To calculate each MRC coefficient a_i, for $3 \leq i \leq n$, Gbolagade-Cotofana MRC requires to perform one subtraction and one multiplication operation. Algorithm 2 requires performing one subtraction, one multiplication and one addition operation. The advantages of the Algorithm 2 include the fact that it does not impose additional restrictions on the RNS moduli set (Gbolagade-Cotofana MRC requires all moduli to be sorted in ascending order).

The best known parallel MRC algorithms, based on lookup tables, have asymptotic complexities in the order of $O(n^2)$ in terms of the required number of tables [15, 16]. In contrast, the proposed algorithm requires only $n - 1$ small lookup tables, each of which has a size of $m_i \log_2 m_i$ bits.

7. CONCLUSION

This paper proposes a new linear time parallel algorithm for magnitude comparison in RNS. It is based on a modification of the MRC-II technique, which allows computing the MRS representation of RNS number using only modulo m_i operations. In further studies this modification can be used for other complicated residue operations, such as division or overflow detection. Other topics for future work include the hardware and software implementations of the proposed algorithms and comprehensive experimental analysis of their performance.

Acknowledgment

The reported study was supported by RFBR, research project No. 16-37-60003 mol_a_dk.

8. REFERENCES

[1] N. S. Szabo and R. I. Tanaka. *Residue Arithmetic and its Application to Computer Technology*. McGraw-Hill, New York, NY, USA, 1967.

[2] B. Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford Univ. Press, New York, NY, USA, 2000.

[3] P. Albicocco, G. C. Cardarilli, A. Nannarelli, and M. Re. Twenty years of research on RNS for DSP: Lessons learned and future perspectives. In *Proc. 14th Int. Symp. Integrated Circuits (ISIC)*, pages 436–439, Singapore, December 2014.

[4] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi. Efficient RNS implementation of elliptic curve point multiplication over $\mathrm{GF(p)}$. *IEEE Trans. VLSI Syst.*, 21(8):1545–1549, August 2013.

[5] Vik Tor Goh and M. U. Siddiqi. Multiple error detection and correction based on redundant residue number systems. *IEEE Trans. Commun.*, 56(3):325–330, March 2008.

[6] G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re. Programmable power-of-two RNS scaler and its application to a QRNS polyphase filter. In *Proc. IEEE Int. Symp. Circuits Syst.*, pages 1102–1105, Kobe, Japan, May 2005.

[7] S. Bi and W. J. Gross. The mixed-radix chinese remainder theorem and its applications to residue comparison. *IEEE Trans. Comput.*, 57(12):1624–1632, December 2008.

[8] G. Dimauro, S. Impedovo, and G. Pirlo. A new technique for fast number comparison in the residue number system. *IEEE Trans. Comput.*, 42(5):608–612, May 1993.

[9] Mi Lu and J.-S. Chiang. A novel division algorithm for the residue number system. *IEEE Trans. Comput.*, 41(8):1026–1032, August 1992.

[10] D. D. Miller, R. E. Altschul, J. R. King, and J. N. Polky. Analysis of the residue class core function of Akushskii, Burcev, and Pak. In M. A. Soderstrand, W. K. Jenkins, G. A Jullien, and F. J. Taylor, Eds. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, pages 390–401. IEEE Press, NJ, USA, 1985.

[11] L. Sousa. Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. In *Proc. 18th IEEE Int. Symp. Comput. Arithmetic*, pages 240–250, Montpellier, France, June 2007.

[12] H. M. Yassine and W. R. Moore. Improved mixed-radix conversion for residue number system architectures. *IEE Proc.-G*, 138(1):120–124, February 1991.

[13] M. Akkal and P. Siy. A new mixed radix conversion algorithm MRC-II. *J. Syst. Arch.*, 53(9):577–586, September 2007.

[14] K. A. Gbolagade and S. D. Cotofana. An $O(n)$ residue number system to mixed radix conversion technique. In *Proc. IEEE Int. Symp. Circuits Syst.*, pages 521–524, Taipei, Taiwan, May 2009.

[15] C. H. Huang. A fully parallel mixed-radix conversion algorithm for residue number applications. *IEEE Trans. Comput.*, C-32(4):398–402, April 1983.

[16] N.B. Chakraborti, J. S. Soundararajan, and A. L. N. Reddy. An implementation of mixed-radix conversion for residue number applications. *IEEE Trans. Comput.*, 35(8):762–764, August 1986.