On Equivalences and Fair Comparisons Among Residue Number Systems with Special Moduli

Behrooz Parhami
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA
parhami@ece.ucsb.edu

Abstract

Properties and applications of residue number systems (RNS) with special moduli of the form $2^k \pm 1$, with a single power-of-2 modulus often also included, have been studied extensively. We show that lack of systematic studies has led to rediscovery of "new" moduli sets that are really equivalent to previously studied ones and that certain comparisons presented to show advantages of some proposed moduli sets are rather unfair. We prove a general mathematical result that allows us to normalize the single power-of-2 modulus, thus removing some of the problematic variations from such proposed sets. We then offer an assessment strategy based on dynamic ranges of the RNS sets being compared, rather than on artificial parameters that may be different for comparable systems.

Keywords— Computer arithmetic, forward converter, modular arithmetic, residue number system, residue-to-binary converter, reverse converter, special RNS moduli.

1. Introduction

Residue number systems (RNS) have been studied for nearly as long as electronic digital computers [Svob59], and are still attracting considerable attention from the research community [Omon07]. They have found niche applications for certain arithmetic-intensive digital signal processing (DSP) problems [Sode86].

In an RNS, a number x is represented by a collection of j residues $(r_1, r_2, r_3, \ldots, r_j)$, where $r_i = x \bmod m_i$, with the m_i parameters constituting a set of j pairwise relatively prime moduli. The dynamic range of RNS$(m_1, m_2, \ldots, m_j)$, that is, the total number of distinct values that it can represent, is $M = m_1 m_2 m_3 \ldots m_j$.

A fundamental operation in computing with RNS representations is that of modular addition. For a general modulus m, modular addition is both more costly and slower than ordinary addition: much slower if addition and modular reduction are done in successive stages. Even a fast implementation, where $x + y$ and $x + y - m$ are computed concurrently and the appropriate one is then chosen as the modular sum (Fig. 1), is slightly slower than ordinary addition, given the presence of a carry-save adder and a multiplexer on the critical path.

Multiplication is another important operation in signal processing applications forming the main target of RNS. Modular multiplication can be performed by interleaving addition and modular reduction steps in a sequential algorithm, thus essentially reducing the problem to multiple modular additions. Alternatively, one can use modular multioperand addition that leads to faster tree multipliers. However, modular multioperand addition is rather costly and slow for an arbitrary modulus m.

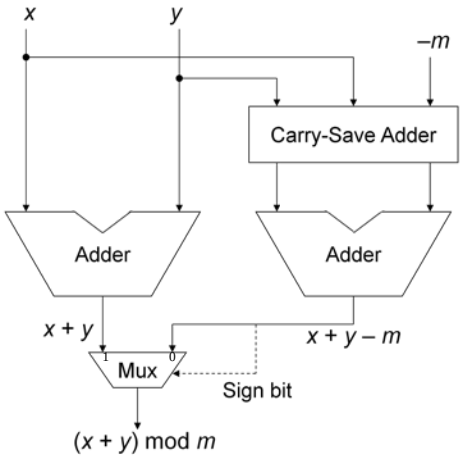

Fig. 1. Fast modular addition for a general modulus *m*.

2. RNS with Special Moduli

Certain special sets of moduli, in which each m_j is of the form 2^k, $2^k - 1$, or $2^k + 1$ lead to simplifications in hardware and reduction of latency for many operations of interest, including binary-to-RNS and RNS-to-binary conversions and the associated forward and reverse hardware converters. Modulo-2^k operations are trivially simple. For example in both 2-operand and multioperand mod-2^k addition, we simply discard any carry that goes into position k (Fig. 2a).

Integers of the form $2^k - 1$ constitute low-cost RNS moduli [Parh76], because as in low-cost arithmetic error codes [Aviz71] they lead to simplified hardware realizations. Mod-m addition for a low-cost modulus m is performed by simply connecting an ordinary adder's carry-out signal to its carry-in line, a scheme known as end-around carry. The equivalent correction in the case of multioperand addition is to reinsert a carry bit that goes to column k into column 0 (Fig. 2b).

Moduli of the form $2^k + 1$, though not as simple to handle as the aforementioned two types (2^k and $2^k - 1$), still lead to efficient hardware realizations by using the diminished-1 encoding (Fig. 3). In the latter encoding, one bit is used to distinguish 0 from nonzero residues. For the latter residues, k bits are used to encode a magnitude that is one less than the true residue. It can be readily proven that an inverted end-around carry performs the necessary correction in this case, in a manner that is similar to the role of end-around carry in mod-$(2^k - 1)$ addition (Fig. 2c).

For these reasons, scores of papers have been written about moduli sets comprised of such special moduli, the attendant algorithms, and hardware implementations for various arithmetic operations [Abda95], [Abda05], [Anan07], [Anan07a], [Conw99], [Hias03], [Tomc08], [Wang00]. Other special moduli, such as those taking the forms $2^k \pm c$, where c is a small constant greater than 1, or $2^k \pm 2^l \pm 1$, for $1 < l < k$, are also of some interest [Hias03a], [Pate07], [Sheu04], but dealing with them is beyond the scope of this paper.

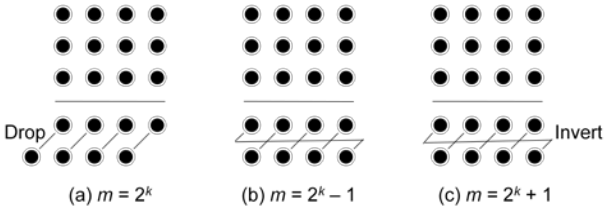

Fig. 2. Modular multioperand addition with special moduli.

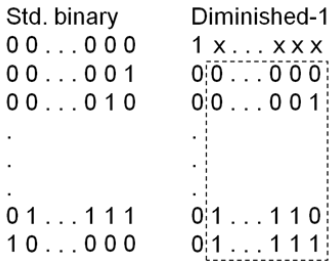

Fig. 3. Standard binary vs. diminished-1 representation of mod-$(2^k + 1)$ residues.

Some special sets of moduli are listed in Table 1 as a small representative sample. The set M4 was proposed in [Anan07a], M1-M3 are sets against which advantages are claimed for M4, and M5 is a set that we use to support our main idea that more systematic assessments and fairer comparisons are needed, both among RNS representations with special moduli and in general. We note that the moduli sets appearing in Table 1 require different widths in their representations, ranging from $3n - 1$ bits for M2 to $4n + 2$ bits for M3. They also offer different dynamic ranges (shown in an approximate manner in the rightmost column of Table 1), ranging from 2^{3n-1} to 2^{4n+1}. Thus, any comparative assessment of implementation cost (VLSI area) and latency as functions of n would be inherently unfair. What we need are cost and latency functions that are expressed in terms of the dynamic range M or another parameter that bears a direct relationship to M.

More specifically, because the dynamic range of an RNS with special moduli of the form considered here is always close to a power of 2, we can take the associated power as a rough indication of the word width in an equivalent binary representation and use it to normalize all comparisons. Put another way, if we start with a desired dynamic range and proceed to choosing one of the moduli sets in Table 1 for the application at hand, different values of n may have to be used. This unfairness in comparison, though quite important, isn't the only problem with the comparative assessments offered in published studies.

Table 1. Some special sets of 3 or 4 RNS moduli.

Set	m_1	m_2	m_3	m_4	Width	Range
M1	$2^n - 1$	2^n	$2^n + 1$	—	$3n + 1$	2^{3n}
M2	$2^n - 1$	2^n	$2^{n-1} - 1$	—	$3n - 1$	2^{3n-1}
M3	$2^n - 1$	2^n	$2^n + 1$	$2^{n+1} - 1$	$4n + 2$	2^{4n+1}
M4	$2^{n+1} - 1$	2^n	$2^n - 1$	—	$3n + 1$	2^{3n+1}
M5	$2^{n+1} - 1$	2^{n+1}	$2^n - 1$	—	$3n + 2$	2^{3n+2}

3. Normalizing the Moduli Set

To treat RNS choices with special moduli methodically, we need some terminology and background.

Definition 1: An RNS with special moduli has a power-of-2 modulus 2^b, plus an arbitrary number of moduli of the form $2^k \pm 1$, that is, it has the moduli set:

$$\{2^k - 1 \mid k = a_1, a_2, ...\} \cup \{2^b\} \cup \{2^k + 1 \mid k = c_1, c_2, ...\}$$

Without loss of generality, we assume:

$$a_1 > a_2 > ... \quad \text{and} \quad c_1 > c_2 > ... \quad \blacksquare$$

We know from number theory that for the low-cost moduli of the form $2^k - 1$ to be pairwise relatively prime, it is necessary and sufficient for the parameters $a_1, a_2, \ldots$ to be pairwise relatively prime. There is, of course, only one parameter b. As for the c_i parameters, defining the subset of moduli of the form $2^k + 1$, selection rules to ensure relative primality are not widely known. If a is odd, then $2^a - 1$ is guaranteed to be relatively prime with respect $2^c + 1$. Only one of the a parameters can be even, in which case it must be factorized to determine limitations on other a_i and c_j values. Numbers of the form $2^c + 1$ can be divided into well-defined classes, with no more than one number allowed from each class [Abda95].

Because in addition and multiplication operations, residues are processed independently and in parallel, operation latency is determined by the dominating parameters a_1, b, and c_1, given that operations with smaller moduli of the same kind are always faster. Let $T(b)$ be the operation latency for the modulus 2^b and assume that T_- and T_+ denote the corresponding latency functions for moduli of the form $2^k - 1$ and $2^k + 1$, respectively. Then, the operation latency for our RNS with special moduli can be written as:

$$T_{\text{RNS}} = \max(T_-(a_1), T(b), T_+(c_1))$$

Operations with a power-of-2 modulus are always simpler than the corresponding operations with moduli of the form $2^k \pm 1$, and operations for low-cost moduli $2^k - 1$ are simpler than corresponding operations for moduli of the form $2^k + 1$, for the same value of k. Thus, we will not lose any performance or efficiency if we assume:

$$c_1 < a_1 \leq b \qquad (*)$$

Using the notation introduced, the defining parameters of the RNS moduli sets of Table 1 are as follows:

M1: $c_1 = a_1 = b$ M2: $a_1 = b$ M3: $c_1 = b < a_1$
M4: $b < a_1$ M5: $a_1 = b$

We see that M1, M3, and M4 do not satisfy the requirements (*) for optimal latency. Consequently, the choice of the corresponding moduli cannot be justified based on the speed of arithmetic operations alone. We thus introduce the following definition.

Definition 2: An RNS with special moduli $\{2^k - 1 \mid k = a_1, a_2, ...\}$ and $\{2^k + 1 \mid k = c_1, c_2, ...\}$, with $a_1 > a_2 > ...$ and $c_1 > c_2 > ...$, and a power-of-2 modulus 2^b is in normal form if $b = \max(a_1, c_1 + 1)$. ■

For a particular set of special moduli, a normal-form RNS maximizes the dynamic range, without any speed penalty, as demonstrated later. Forward and reverse conversion latencies are discussed next.

In converting from RNS to binary, certain methods require that the multiplicative inverse of one modulus with respect to another modulus be found and multiplied by other terms, with the results then summed up.

Example 1: Inverses for M4 of Table 1.

$$X_A = (2^n)^{-1} \bmod (2^{n+1} - 1) = 2$$
$$X_B = (2^n)^{-1} \bmod (2^n - 1) = 1$$
$$X_C = (2^n - 1)^{-1} \bmod (2^{n+1} - 1) = -2$$
$$X_D = (2^{n+1} - 1)^{-1} \bmod (2^n - 1) = 1$$

These results can be readily confirmed. For example, 2 is the mod-$(2^{n+1} - 1)$ inverse of 2^n because $2 \times 2^n = 1 \bmod (2^{n+1} - 1)$. Clearly, 2, 1, and –2 are quite easily multiplied by other quantities, because the product can be derived by mere shifting and complementation. ■

Example 2: Inverses for M5 of Table 1.

$$X_A' = (2^{n+1})^{-1} \bmod (2^{n+1} - 1) = 1$$
$$X_B' = (2^{n+1})^{-1} \bmod (2^n - 1) = 2$$

We also have $X_C' = X_C$ and $X_D' = X_D$, as in Example 1. The moduli set M5 has a range that is double that of M4, while its widest residue has the same width. Given that speed of arithmetic in RNS is dictated by the width of the widest residue, M4 with its more limited range offers no advantage over M5 that could justify its use. ■

It is seen from Examples 1 and 2 that the multiplicative inverses of M5's moduli are as simple as those for M4, producing identical circuit speedups and simplifications. Similar results can be obtained for all other multiplicative inverses reported in the literature. To generalize the discussion above, we need the following definition.

Definition 3: The (arithmetic) weight of an integer x is the minimum number of powers of 2, with positive or negative signs, that add up to x. So, the arithmetic weight of $\pm 2^i$ is 1, while that of $\pm(2^j \pm 2^k)$, $j \neq k$, is 2. ■

Multiplying y by a weight-1 integer x requires only shifting and possibly complementation. In the case of a weight-2 multiplier x, a single addition is also needed. The one extra addition in the latter case may not affect the critical path length if it is merged into a multioperand addition process. For example, adding 6 integers is not any slower than adding 5 items.

We now have all the tools and concepts needed to present the following general result.

Theorem 1: Changing the modulus 2^b in the RNS of Definition 1 to 2^{b+1} increases the weight of the inverses of moduli with respect to each other by at most 1.

Proof: First, consider the inverse of the power-of-2 modulus. In this case, the following more general results can be proven that show the weight of the inverse to be 1 (the inverse is a positive or negative power of 2):

$$(2^i)^{-1} \bmod (2^j - 1) = 2^{(-i) \bmod j}$$
$$(2^i)^{-1} \bmod (2^j + 1) = \pm 2^{(-i) \bmod j}$$

In the latter equality, the sign depends on the parity of l, where $i = lj + (i \bmod j)$. Next, let $(2^a - 1)^{-1} \bmod 2^b = x$, with x of weight w; the final case of $(2^c + 1)^{-1} \bmod 2^b$ is similar and is thus omitted in this proof. Let $(2^a - 1)^{-1} \bmod 2^{b+1} = y$. The fact that $x(2^a - 1) = 1 \bmod 2^b$ implies $x(2^a - 1) = h2^b + 1$ for some integer h. It is readily verified that for h even, we have $y = x$, while for h odd, $y = x + 2^b$, thus establishing the desired result. ■

Based on Theorem 1, doubling the power-of-2 modulus in a special set of moduli either has no effect on the simplicity of modular inverse computations or has a very small effect on cost and latency. We can thus modify the M3 and M4 example systems in Table 1 by doubling their power-of-2 moduli without an adverse effect on the speed of arithmetic. Once we have done this, we normalize the power-of-2 modulus to be 2^n in all cases, to get the equivalent normalized RNSs in Table 2.

As shown in Table 3, in normalizing an RNS with special moduli, we sometimes increase the value of the parameter n required to achieve a desired dynamic range. However, as noted earlier, it is not the value of the parameter n, but rather the latency of the slowest RNS channel that determines the overall latency.

Table 2. Three RNSs from Table 1, in normalized form.

Set	m_1	m_2	m_3	m_4	Width	Range
M1′	$2^n - 1$	2^n	$2^{n-1} + 1$	—	$3n$	2^{3n-1}
M3′	$2^{n-1} - 1$	2^n	$2^{n-1} + 1$	$2^n - 1$	$4n - 1$	2^{4n-2}
M4′	$2^n - 1$	2^n	$2^{n-1} - 1$	—	$3n - 1$	2^{3n-1}

Table 3. Some specific RNSs with special moduli.

Equiv. range (b)	12	16	20	24	28	32
n for M1/M1′	3/4	6/6	7/7	8/9	10/10	11/11
n for M3/M3′	3/4	4/5	5/6	6/7	7/8	8/9
n for M4/M4′	4/5	5/6	7/7	8/9	9/10	11/11

4. Implications of Our Results

Our results find applications in selecting special moduli for an RNS with a desired dynamic range, so as to minimize the operation latencies. Of course, latency is not the only consideration and we must also take into account implementation cost (VLSI area) and power consumption. Let us focus on area and take power to be roughly proportional to area. The moduli $2^k - 1$, 2^k, and $2^k + 1$ have nearly identical contributions to the dynamic range. It thus makes sense to use larger moduli of the form $2^k + 1$ only when an equivalent range extension cannot be achieved by raising the other moduli, especially 2^k. We are thus led to the following (this is an informal result, rather than a theorem, because modification of the moduli must be done with an eye toward keeping them pairwise relatively prime and may thus entail some trial and error).

Assertion: If the conditions (*) are not satisfied for an RNS with special moduli, then a lower latency or wider range can always be achieved by changing the moduli, that is, modifying the parameters a_1, b, and c_1. ■

A final observation is in order here. Comparing M5 with M2 of Table 1, we note that the former is in fact M2, but with the parameter n replaced by $n + 1$. In other words, M5 is nothing but an instance of M2. We are thus led to the conclusion that the moduli set M4 proposed and evaluated in [Anan07a] is neither novel nor offers advantages over the already proposed moduli sets that have been extensively studied in the literature on computer arithmetic and DSP.

5. Conclusion

We have pointed out certain features of special RNS moduli of the form $2^k \pm 1$, with one power-of-2 modulus, exposing some equivalences and redundancies of efforts in the process. There do exist other classes of special moduli that can be studied separately or in combination with moduli of the form $2^k \pm 1$. One interesting example is moduli of the form β^k for different values of β and k, which lead to simplifications if radix-β_i representation is used in the ith RNS channel [Pali00].

References

[Abda95] M. Abdallah and A. Skavantzos, "New Multi-Moduli Residue and Quadratic Residue Systems for Large Dynamic Ranges," *Proc. 29th Asilomar Conf. Signals, Systems, and Computers*, 1995, pp. 961-965.

[Abda05] M. Abdallah and A. Skavantzos, "On Multi Moduli Residue Number Systems with Moduli of Forms r^a, $r^b - 1$, $r^c + 1$," *IEEE Trans. Circuits and Systems I*, Vol. 52, No. 7, pp. 1253-1266, 2005.

[Anan07] P. V. Ananda Mohan and A. B. Premkumar, "RNS-to-Binary Converters for Two Four-Moduli Sets $\{2^n–1, 2^n, 2^n+1, 2^{n+1}–1\}$ and $\{2^n–1, 2^n, 2^n+1, 2^{n+1}+1\}$," *IEEE Trans. Circuits and Systems I*, Vol. 54, No. 6, pp. 1245-1254, 2007.

[Anan07a] P. V. Ananda Mohan, "RNS-to-Binary Converter for a New Three-Moduli Set $\{2^{n+1}–1, 2^n, 2^n–1\}$," *IEEE Trans. Circuits and Systems II*, Vol. 54, No. 9, pp. 775-779, 2007.

[Aviz71] A. Avizienis, "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design," *IEEE Trans. Computers*, Vol. 20, No. 11, pp. 1322-1331, 1971.

[Bi88] G. Bi and E. V. Jones, "Fast Conversion Between Binary and Residue Numbers," *Electronics Letters*, Vol. 24, No. 19, pp. 1195-1197, 1988.

[Cao07] B. Cao, C. H. Chang, and T. Srikanthan, "A Residue-to-Binary Converter for a New Five-Moduli Set," *IEEE Trans. Circuits and Systems I*, Vol. 54, No. 5, pp. 1041-1049, 2007.

[Conw99] R. Conway and J. Nelson, "Fast Converter for 3 Moduli RNS Using New Property of CRT," *IEEE Trans. Computers*, Vol. 48, No. 8, pp. 852-860, 1999.

[Conw03] R. Conway and J. Nelson, "New CRT-Based RNS Converter Using Restricted Moduli Set," *IEEE Trans. Computers*, Vol. 52, No. 5, pp. 572-578, 2003.

[Conw04] R. Conway and J. Nelson, "Improved RNS FIR Filter Architectures," *IEEE Trans. Circuits and Systems II*, Vol. 51, No. 1, pp. 26-28, 2004.

[Gall97] D. Gallaher, F. E. Petry, and P. Srinivasan, "The Digit Parallel Method for Fast RNS to Weighted Number System Conversion for Specific Moduli $(2^k – 1, 2^k, 2^k + 1)$," *IEEE Trans. Circuits and Systems II*, Vol. 44, No. 1, pp. 53-57, 1997.

[Hias98] A. A. Hiasat and H. S. Abdel-Aty-Zohdy, "Residue to Binary Arithmetic Converter for the Moduli Set $(2^k, 2^k – 1, 2^{k–1} – 1)$," *IEEE Trans. Circuits and Systems II*, Vol. 45, No. 2, pp. 204-209, 1998.

[Hias03] A. Hiasat and A. Sweiden, "Residue Number System to Binary Converter for the Moduli Set $(2^{n–1}, 2^n – 1, 2^n + 1)$," *J. Systems Architecture*, Vol. 49, Nos. 1-2, pp. 53-58, 2003.

[Hias03a] A. Hiasat, "An Arithmetic Residue to Binary Conversion Technique," *Integration, the VLSI J.*, Vol. 36, Nos. 1-2, pp. 13-25, 2003.

[Mola10] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi, "Efficient Reverse Converter Designs for the New 4-Moduli Sets $\{2^n–1, 2^n, 2^n+1, 2^{2n+1}–1\}$ and $\{2^n–1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ Based on New CRTs," *IEEE Trans. Circuits and Systems I*, Vol. 57, No. 4, pp. 823-835, 2010.

[Omon07] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*, World Scientific, 2007.

[Pali00] V. Paliouras and T. Stouraitis, "Novel High-Radix Residue Number System Architectures," *IEEE Trans. VLSI*, Vol. 8, No. 3, pp. 1059-1073, 2000.

[Parh76] B. Parhami, "Low-Cost Residue Number Systems for Computer Arithmetic," *AFIPS Conf. Proc.*, Vol. 45 (1976 National Computer Conf.), AFIPS Press, pp. 951-956.

[Pate07] R. A. Patel, M. Benaissa, and S. Boussakta, "Fast Modulo $2^n – (2^{n–2} + 1)$ Addition: A New Class of Adders for RNS," *IEEE Trans. Computers*, Vol. 56, No. 4, pp. 572-576, 2007.

[Pies95] S. J. Piestrak, "A High-Speed Realization of Residue to Binary System Converter," *IEEE Trans. Circuits and Systems II*, Vol. 42, No. 12, pp. 661-663, 1995.

[Sheu04] M.-H. Sheu, S.-H. Lin, C. Chen, and S.-W. Yang, "An Efficient VLSI Design for a Residue to Binary Converter for General Balance Moduli $(2^n – 3, 2^n + 1, 2^n – 1, 2^n + 3)$," *IEEE Trans. Circuits and Systems II*, Vol. 51, No. 3, pp. 152-155, 2004.

[Sode86] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor (eds.), *Residue Number System Arithmetic*, IEEE Press, 1986.

[Svob59] A. Svoboda, "The Numerical System of Residual Classes in Mathematical Machines," in *Information Processing* (Proc. UNESCO Conf., 1959), pp. 419-422.

[Tomc08] T. Tomczak, "Fast Sign Detection for RNS $(2^n–1, 2^n, 2^n+1)$," *IEEE Trans. Circuits and Systems I*, Vol. 55, No. 6, pp. 1502-1511, 2008.

[Wang99] Y. Wang, M. N. S. Swamy, and M. O. Ahmad, "Residue to Binary Number Converters for Three Moduli Sets," *IEEE Trans. Circuits and Systems II*, Vol. 46, No. 2, pp. 180-183, 1999.

[Wang00] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang, "A High-Speed Residue-to-Binary Converter for Three-Moduli $(2^k, 2^k – 1, 2^{k–1} – 1)$ RNS and a Scheme for Its VLSI Implementation," *IEEE Trans. Circuits and Systems II*, Vol. 47, No. 12, pp. 1576-1581, 2000.

[Wang00a] Z. Wang, G. A. Jullien, and W. C. Miller, "An Improved Residue to Binary Converter," *IEEE Trans. Circuits and Systems I*, Vol. 47, No. 9, pp. 1437-1440, 2000.