# Low-cost residue number systems for computer arithmetic

*by* BEHROOZ PARHAMI
*Arya-Mehr University of Technology*
Tehran, Iran

## ABSTRACT

The representation of integers by their residues with respect to a set of pairwise-prime moduli is known as the residue number representation system and has been shown to have several advantages over conventional number systems for digital computers. In this paper, residue systems are considered for which each modulus is of the form $2^b$-1. Such systems result in relatively high storage efficiency as well as simple algorithms for addition, subtraction multiplication, conversion, and reconversion; hence the name "low-cost." The question of existence for low-cost residue number systems is examined. It is shown that the additional storage requirement with respect to binary representation is at most one bit per word. Guidelines are given for optimal selection of the set of moduli to represent a desired range of integers. Algorithms for various operations in a low-cost residue system are described.

## INTRODUCTION

When dealing with large numbers in digital computers, the computations are slowed down because of the requirement for carry or borrow propagation through many stages of logic in addition and subtraction operations and for long iterative algorithms to perform multiplication and division. Attempts to eliminate the propagation of carries and borrows have resulted in proposals for stored-carry[1] and signed-digit[2] number representation systems. The residue number system[3] does not totally eliminate carry propagation but limits it to within a few stages by representing large numbers as an ordered set of smaller numbers that can be processed independently and in parallel. This is particularly advantageous in multiplication which becomes almost as simple and as fast as addition. However, the complexity of division in residue number systems makes them unsuitable for general-purpose use.

In this paper, residue number systems are reviewed briefly and their properties are enumerated. A class of residue number representation systems which results in relatively high storage efficiency as well as simple algorithms for addition, subtraction, multiplication, conversion, and reconversion algorithms is introduced. The questions of existence, selection, storage efficiency, and algorithms for such "low-cost" residue systems are examined. The storage requirement for each word is shown to be within one bit of the binary representation. Algorithms needed for basic operations and conversions are discussed.

## RESIDUE NUMBER SYSTEMS

A *residue number system*[4,5] is one in which a numerical value n is represented by a k-tuple whose components are the residues of n with respect to an ordered set of k moduli

$$p = \langle p_1, p_2, \ldots, p_k \rangle \tag{1}$$

which are relatively prime pairwise. Hence, n is represented by the k-tuple

$$r = \langle r_1, r_2, \ldots, r_k \rangle \tag{2}$$

such that

$$r_i = p_i | n \; ; \; i = 1, 2, \ldots, k \tag{3}$$

where $\rho = \mu | \xi$ means that $\rho$ is the smallest non-negative integer satisfying $\xi = \rho + \beta\mu$ for some integer $\beta$. The *range* of a residue system (i.e., the number of distinct values representable) is:

$$N = \prod_{i=1}^{k} p_i \tag{4}$$

To represent a negative integer $-n$, we simply represent the positive integer $N-n$ since we have

$$p_i | (N-n) = p_i | (-n) \; ; \; i = 1, 2, \ldots, k \tag{5}$$

The integer $N-n$ is the *additive inverse* of n and is denoted by $\bar{n}$. The residue representation $\bar{r}$ of $\bar{n}$ has the following relation with the representation r of n:

$$\bar{r}_i = p_i | (p_i - r_i) \; ; \; i = 1, 2, \ldots, k \tag{6}$$

If binary representation is used for the residues, the number of bits required for storing each value in the residue system is

$$B = \sum_{i=1}^{k} b_i = \sum_{i=1}^{k} \lceil \log_2 p_i \rceil \tag{7}$$

which is always greater than or equal to $\lceil \log_2 N \rceil$, the number of bits needed for the binary representation of N distinct values. Hence, a residue number system is less efficient than the binary representation in terms of storage space.

Addition and multiplication in a residue system are done by performing the corresponding operation (modulo $p_i$) on the i-th residues of the two numbers, independently of other residues. Hence, showing the sum and product of $x = <x_1, x_2, \ldots, x_k>$ and $y = <y_1, y_2, \ldots, y_k>$ by s and m, respectively, we can write:

$$s_i = p_i | (x_i + y_i) \; ; \; i = 1,2, \ldots, k \tag{8}$$

$$m_i = p_i | (x_i \cdot y_i) \; ; \; i = 1,2, \ldots, k \tag{9}$$

Subtraction is performed by adding the additive inverse, as defined by (6). Thus, the carry propagation delay for addition and subtraction is reduced and the construction of very fast multiplication circuits is made possible. However, comparison of magnitudes, and hence division, and also detection of overflow conditions are fairly complex in residue number systems. Hence, such systems are not suitable for general-purpose use.

To find the normal representation n of a residue number $r = <r_1, r_2, \ldots, r_k>$, the following equation may be used

$$n = N | \sum_{i=1}^{k} \left( (r_i c_i \frac{N}{p_i} \right) \tag{10}$$

where the coefficient $c_i$ is selected to be the smallest integer satisfying

$$c_i = p_i (1 + \beta_i p_i) / N \tag{11}$$

for some integer $\beta_i$.

Another reconversion process uses the following algorithm which is a formalization of the procedure given in Reference 5.

*Algorithm R* (12)

[1] $v \leftarrow r; n \leftarrow 0; w \leftarrow 1; u_j \leftarrow 1, j = 1,2, \ldots, k; i \leftarrow k$

[2] Find smallest integer d such that for some $\beta$,
$d = (v_i + \beta p_i) / u_i$

[3] $n \leftarrow n + wd$

[4] For $j = 1,2, \ldots, k$ set $v_j \leftarrow p_j | (v_j - u_j d)$ and
$u_j \leftarrow p_j | (u_j p_i)$

[5] $w \leftarrow wp_i$

[6] $i \leftarrow i - 1$

[7] if $i > 0$ then go to Step [2] else stop

## LOW-COST RESIDUE SYSTEMS

A residue number representation system is *low-cost* if each modulus $p_i$ is selected such that:

$$p_i = 2^{b_i} - 1 \; ; \; b_i > 2 \tag{13}$$

The name "low-cost" is justified because of the relatively high storage efficiency and the simplicity of addition subtraction, multiplication, conversion, and reconversion algorithms as will be seen in the remainder of this paper. In this section, we will only concentrate on the existence of such systems and their storage efficiency.

The selection of $b_i$'s must be made such that the resulting $p_i$'s are pairwise prime. It can be proven that $p_i$ and $p_j$ are relatively prime if and only if the corresponding $b_i$ and $b_j$ are relatively prime (see Theorem 1 in the Appendix). Using this result, Table I has been constructed to show the maximal sets of pairwise-prime $b_i$'s for $b_i \leq 20$, since making $b_i$ larger than 20 may defeat the advantage of residue number systems in breaking long numbers into several short components. We define, as a measure of this advantage, the *dissection factor:*

$$\delta = \max_i (b_i) / \Sigma_i b_i \tag{14}$$

Table II shows possible selections of $b_i$'s for a given total number of bits, B, which satisfy the following criteria, in the order given: (1) Minimum value of $\delta$, and (2) Smallest number of $b_i$'s. The second criterion is justified by the fact that once the size of the longest group is fixed at its minimum value, no speed advantage results from making the other groups shorter. Figure 1 shows the same results graphically.

We define as a measure of storage efficiency, the ratio of N to the range of the binary system with the same number of bits:

$$n = N/2^B = \prod_{i=1}^{k} (p_i / 2^{b_i}) = \prod_{i=1}^{k} (1 - 2^{-b_i}) \tag{15}$$

It can be proven (see Theorem 2 in the Appendix) that

TABLE I—Maximal Compatible Sets of $b_i$'s ($b_i \leq 20$) for Low-Cost Residue Number Systems

| Set No. | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $b_7$ | $b_8$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 |
| 2 | 2 | 5 | 7 | 9 | 11 | 13 | 17 | 19 |
| 3 | 2 | 7 | 11 | 13 | 15 | 17 | 19 | |
| 4 | 3 | 4 | 5 | 7 | 11 | 13 | 17 | 19 |
| 5 | 3 | 5 | 7 | 8 | 11 | 13 | 17 | 19 |
| 6 | 3 | 5 | 7 | 11 | 13 | 16 | 17 | 19 |
| 7 | 3 | 5 | 11 | 13 | 14 | 17 | 19 | |
| 8 | 3 | 7 | 10 | 11 | 13 | 17 | 19 | |
| 9 | 3 | 7 | 11 | 13 | 17 | 19 | 20 | |
| 10 | 4 | 5 | 7 | 9 | 11 | 13 | 17 | 19 |
| 11 | 4 | 7 | 11 | 13 | 15 | 17 | 19 | |
| 12 | 5 | 6 | 7 | 11 | 13 | 17 | 19 | |
| 13 | 5 | 7 | 8 | 9 | 11 | 13 | 17 | 19 |
| 14 | 5 | 7 | 9 | 11 | 13 | 16 | 17 | 19 |
| 15 | 5 | 7 | 11 | 12 | 13 | 17 | 19 | |
| 16 | 5 | 7 | 11 | 13 | 17 | 18 | 19 | |
| 17 | 5 | 9 | 11 | 13 | 14 | 17 | 19 | |
| 18 | 7 | 8 | 11 | 13 | 15 | 17 | 19 | |
| 19 | 7 | 9 | 10 | 11 | 13 | 17 | 19 | |
| 20 | 7 | 9 | 11 | 13 | 17 | 19 | 20 | |
| 21 | 7 | 11 | 13 | 15 | 16 | 17 | 19 | |
| 22 | 11 | 13 | 14 | 15 | 17 | 19 | | |

TABLE II—Best Choices for $b_i$'s in a Low-Cost Residue Number System with a Given Total Number of Bits (B)

| B | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $\delta(\%)$ |
|---|---|---|---|---|---|---|
| 5 | 2 | 3 | | | | 60.0 |
| 7 | 3 | 4 | | | | 57.1 |
| 8 | 3 | 5 | | | | 62.5 |
| 9 | 4 | 5 | | | | 55.6 |
| 10 | 2 | 3 | 5 | | | 50.0 |
| 11 | 5 | 6 | | | | 54.5 |
| 12 | 3 | 4 | 5 | | | 41.7 |
| 13 | 6 | 7 | | | | 53.8 |
| 14 | 2 | 5 | 7 | | | 50.0 |
| | 3 | 4 | 7 | | | |
| 15 | 3 | 5 | 7 | | | 46.7 |
| 16 | 4 | 5 | 7 | | | 43.8 |
| 17 | 2 | 3 | 5 | 7 | | 41.2 |
| 18 | 5 | 6 | 7 | | | 38.9 |
| 19 | 3 | 4 | 5 | 7 | | 36.8 |
| 20 | 5 | 7 | 8 | | | 40.0 |
| 21 | 5 | 7 | 9 | | | 42.9 |
| 22 | 5 | 8 | 9 | | | 40.9 |
| 23 | 3 | 5 | 7 | 8 | | 34.8 |
| 24 | 7 | 8 | 9 | | | 37.5 |
| 25 | 4 | 5 | 7 | 9 | | 36.0 |
| 26 | 7 | 9 | 10 | | | 38.5 |
| 27 | 7 | 9 | 11 | | | 40.7 |
| 28 | 7 | 10 | 11 | | | 39.3 |
| | 8 | 9 | 11 | | | |
| 29 | 5 | 7 | 8 | 9 | | 31.0 |
| 30 | 9 | 10 | 11 | | | 36.7 |
| 31 | 3 | 7 | 10 | 11 | | 35.5 |
| | 4 | 7 | 9 | 11 | | |
| | 5 | 7 | 8 | 11 | | |
| 32 | 5 | 7 | 9 | 11 | | 34.4 |
| 33 | 5 | 8 | 9 | 11 | | 33.3 |
| 34 | 2 | 5 | 7 | 9 | 11 | 32.4 |
| | 3 | 5 | 7 | 8 | 11 | |

| B | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $\delta(\%)$ |
|---|---|---|---|---|---|---|---|
| 35 | 7 | 8 | 9 | 11 | | | 31.4 |
| 36 | 4 | 5 | 7 | 9 | 11 | | 30.6 |
| 37 | 7 | 9 | 10 | 11 | | | 29.7 |
| 38 | 5 | 9 | 11 | 13 | | | 34.2 |
| 39 | 7 | 8 | 11 | 13 | | | 33.3 |
| | 7 | 9 | 10 | 13 | | | |
| 40 | 5 | 7 | 8 | 9 | 11 | | 27.5 |
| 41 | 5 | 11 | 12 | 13 | | | 31.7 |
| | 7 | 10 | 11 | 13 | | | |
| | 8 | 9 | 11 | 13 | | | |
| 42 | 2 | 7 | 9 | 11 | 13 | | 31.0 |
| | 3 | 7 | 8 | 11 | 13 | | |
| | 4 | 5 | 9 | 11 | 13 | | |
| | 5 | 6 | 7 | 11 | 13 | | |
| | 5 | 7 | 8 | 9 | 13 | | |
| 43 | 7 | 11 | 12 | 13 | | | 30.2 |
| | 9 | 10 | 11 | 13 | | | |
| 44 | 3 | 7 | 10 | 11 | 13 | | 29.5 |
| | 4 | 7 | 9 | 11 | 13 | | |
| | 5 | 7 | 8 | 11 | 13 | | |
| 45 | 5 | 7 | 9 | 11 | 13 | | 28.9 |
| 46 | 5 | 8 | 9 | 11 | 13 | | 28.3 |
| 47 | 2 | 5 | 7 | 9 | 11 | 13 | 27.7 |
| | 3 | 5 | 7 | 8 | 11 | 13 | |
| 48 | 5 | 7 | 11 | 12 | 13 | | 27.1 |
| | 7 | 8 | 9 | 11 | 13 | | |
| 49 | 4 | 5 | 7 | 9 | 11 | 13 | 26.6 |
| 50 | 7 | 9 | 10 | 11 | 13 | | 26.0 |
| 51 | 7 | 13 | 15 | 16 | | | 31.4 |
| 52 | 5 | 9 | 11 | 13 | 14 | | 26.9 |
| 53 | 5 | 7 | 8 | 9 | 11 | 13 | 24.5 |
| 54 | 7 | 8 | 11 | 13 | 15 | | 27.8 |
| 55 | 11 | 13 | 15 | 16 | | | 29.1 |
| 56 | 7 | 9 | 11 | 13 | 16 | | 28.6 |

| B | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $\delta(\%)$ |
|---|---|---|---|---|---|---|---|
| 57 | 11 | 13 | 16 | 17 | | | 29.8 |
| | 11 | 14 | 15 | 17 | | | |
| 58 | 2 | 11 | 13 | 15 | 17 | | 29.3 |
| | 3 | 11 | 13 | 14 | 17 | | |
| | 5 | 7 | 13 | 16 | 17 | | |
| | 5 | 9 | 11 | 16 | 17 | | |
| | 5 | 9 | 13 | 14 | 17 | | |
| | 5 | 11 | 12 | 13 | 17 | | |
| | 7 | 8 | 11 | 15 | 17 | | |
| | 7 | 10 | 11 | 13 | 17 | | |
| | 8 | 9 | 11 | 13 | 17 | | |
| 59 | 11 | 15 | 16 | 17 | | | 28.8 |
| | 13 | 14 | 15 | 17 | | | |
| 60 | 3 | 11 | 13 | 16 | 17 | | 28.3 |
| | 4 | 11 | 13 | 15 | 17 | | |
| | 5 | 9 | 13 | 16 | 17 | | |
| | 5 | 11 | 13 | 14 | 17 | | |
| | 7 | 8 | 13 | 15 | 17 | | |
| | 7 | 9 | 11 | 16 | 17 | | |
| | 7 | 11 | 12 | 13 | 17 | | |
| | 9 | 10 | 11 | 13 | 17 | | |
| 61 | 5 | 7 | 9 | 11 | 13 | 16 | 26.2 |
| 62 | 7 | 11 | 13 | 15 | 16 | | 25.8 |
| 63 | 7 | 11 | 13 | 15 | 17 | | 27.0 |
| 64 | 7 | 11 | 13 | 16 | 17 | | 26.6 |
| | 8 | 11 | 13 | 15 | 17 | | |
| | 9 | 11 | 13 | 14 | 17 | | |
| 65 | 2 | 7 | 11 | 13 | 15 | 17 | 26.2 |
| | 3 | 5 | 11 | 13 | 16 | 17 | |
| | 5 | 7 | 9 | 11 | 16 | 17 | |
| | 5 | 7 | 11 | 12 | 13 | 17 | |
| | 7 | 8 | 9 | 11 | 13 | 17 | |

for any low-cost residue number system $0.5<\eta<1$ from which we can conclude

$$2^{B-1}<N<2^B \qquad (16)$$

This shows that the storage requirement for a low-cost residue system is within one bit of the most efficient representation. It also shows that N is an increasing function of B.

To select a low-cost residue system, B must be determined first. To do this, we first note that among all choices for the set of moduli for each value of B, given by Table II, the one for which $\min_i(b_i)$ is a maximum results in the largest possible range (see Theorem 3 in the Appendix). If more than one set has this maximum value for $\min_i(b_i)$, we look at the second smallest $b_i$ in the sets, etc. Table III gives the maximum range obtainable for each value of B. Since, in a low-cost residue system, the storage requirement is dictated by B and the processing speed by $\max_i(b_i)$, the final choice for B among the values which provide adequate range may involve a tradeoff between these two factors. For example if B=51 is sufficient for some desired range, B=52 and B=53 must also be considered for the final selection, since they provide higher processing speeds at the expense of more storage space.

## LOW-COST ALGORITHMS

We first note that in dealing with numbers represented in a residue system, the following operations in-

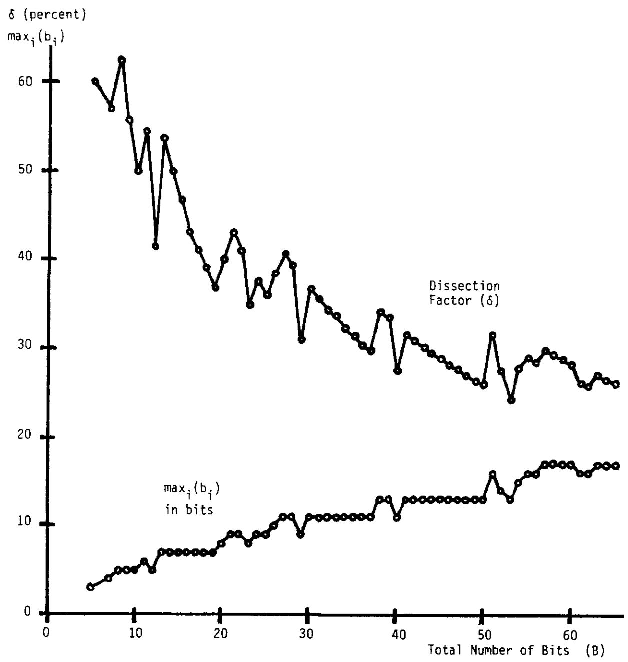


Figure 1—The values of $\max_i(b_i)$ and $\delta$ as functions of B

volving the set of moduli p are required (numbers following each operation show the equations where it is used):

1. Subtraction from $p_i$ : (6)
2. Addition modulo $p_i$ : (8)
3. Determination of residues with respect to $p_i$ : (3), (6), (12)
4. Multiplication modulo $p_i$ : (9)
5. Multiplication by $p_i$ : (12)
6. Division by $p_i$ : (10)

We will show that low-cost algorithms exist for performing all of the above operations in a low-cost residue number system. Here, the term "low-cost" refers to the computer implementation of algorithms, keeping in mind that in digital computers addition and subtraction are the fastest and least expensive of the four basic operations, while division is the slowest and most expensive to implement. Most of the operations to be described are also used in encoding, decoding and arithmetic operations for low-cost arithmetic error codes.[6,7]

Subtraction of a $b_i$-bit binary number x from $p_i$ is quite simple since $p_i=2^{b_i}-1$ is represented in binary as $b_i$ digits of 1. Hence, the digits of $p_i-x$ are the logical complements of the digits of x.

Addition of two $b_i$-bit binary numbers modulo $p_i$ is also simple. It consists of a simple $b_i$-bit binary addition with end-around carry; i.e., the carry generated by the last digit position is inserted into the first digit position. This is true since for a sum which is greater than $p_i$, we have to subtract $p_i=2^{b_i}-1$ in order to obtain its modulo $p_i$ residue. This is done by subtracting $2^{b_i}$ (discarding the outgoing carry) and adding 1 (inserting a carry into the first digit position). The only problem arises when the sum is equal to $p_i$, in which case we either need special circuitry to detect this condition and insert a carry into the first digit position if it arises, or simply leave the result as it is and have two representations for zero. This latter approach will cause no difficulty since in all modulo-$p_i$ operations the two values 0 and $2^{b_i}-1$ are entirely equivalent.

To determine the residue of a binary number x with respect to $p_i$, we simply break x into $b_i$-bit bytes, starting at the right end, and add the resulting bytes modulo $p_i$. This is true since the residue of $2^{b_i}$ with respect to $p_i$ is equal to 1 and the value of x is a polynomial in $2^{b_i}$, with the values of the $b_i$-bit bytes of x as the coefficients. Hence the residue of x with respect to $p_i$ is the same as the residue of the sum of these coefficients with respect to $p_i$, which is the modulo $p_i$ addition of these coefficients.

Multiplication in digital computers is usually performed through multiple additions, either sequentially by a single adder or in parallel by using a number of carry-save adders.[8] Hence, modulo-$p_i$ multiplication of two numbers can be performed through a number of modulo-$p_i$ additions, the algorithm for which was discussed previously.

Multiplication of a binary number x by $p_i=2^{b_i}-1$ can be done by a single subtraction $x.2^{b_i}-x$, since $x.2^{b_i}$ can be easily obtained through shifting x to the left by $b_i$ bits (inserting $b_i$ zero to the right of x).

Finally, division by $p_i$ (of a number which is a multiple of $p_i$) can be done by a very interesting algorithm[7] which is obtained by observing that $x=x.2^{b_i}-x.p_i$. Now, the first $b_i$ bits of $x.2^{b_i}$ are known to be zero and since we have $x.p_i$, the first $b_i$ bits of x can be obtained by subtraction. These $b_i$ bits of x now form the second $b_i$ bits of $x.2^{b_i}$ and, hence, the second $b_i$ bits of x are obtained by another subtraction, taking into account a borrow which may have been generated by the first subtraction. This process is continued until all the digits of x are computed.

## CONCLUSION

In this paper, we have introduced the class of low-cost residue number representation systems and studied their properties. It appears that such systems alleviate the storage inefficiency normally associated with residue number systems and simplify many of the basic algorithms. The division process, however, remains complex. Therefore, such systems are useful only for special applications.

One disadvantage of the low-cost residue number system is that the moduli, and hence the residues, are larger than those for conventional residue systems with no restriction on $p_i$'s. Therefore, carry propagation delay is not reduced by as much. However, this

TABLE III—Maximum Range of Low-Cost Residue Number Systems with a Given B and with Minimum $\delta$

| B | max N | log (max N) | B | max N | log (max N) |
|---|---|---|---|---|---|
| 5 | 21 | 1.322 | 36 | 61772533935 | 10.791 |
| 7 | 105 | 2.021 | 37 | 135899574657 | 11.133 |
| 8 | 217 | 2.336 | 38 | 265605682657 | 11.424 |
| 9 | 465 | 2.667 | 39 | 543797467521 | 11.735 |
| 10 | 651 | 2.814 | 40 | 1050133076895 | 12.021 |
| 11 | 1953 | 3.291 | 41 | 2184820937985 | 12.339 |
| 12 | 3255 | 3.513 | 42 | 4202071339935 | 12.623 |
| 13 | 8001 | 3.903 | 43 | 8764987527681 | 12.943 |
| 14 | 13335 | 4.125 | 44 | 16832955054495 | 13.226 |
| 15 | 27559 | 4.440 | 45 | 33731921697439 | 13.528 |
| 16 | 59055 | 4.771 | 46 | 67729449077535 | 13.831 |
| 17 | 82677 | 4.917 | 47 | 117830685381465 | 14.071 |
| 18 | 248031 | 5.395 | 48 | 277472259124095 | 14.443 |
| 19 | 413385 | 5.616 | 49 | 505978825461585 | 14.704 |
| 20 | 1003935 | 6.002 | 50 | 1113153416015487 | 15.047 |
| 21 | 2011807 | 6.304 | 51 | 2233832636833665 | 15.349 |
| 22 | 4039455 | 6.606 | 52 | 4351417898969631 | 15.639 |
| 23 | 7027545 | 6.847 | 53 | 8601640032846945 | 15.935 |
| 24 | 16548735 | 7.219 | 54 | 17792433492601215 | 16.250 |
| 25 | 30177105 | 7.480 | 55 | 35442210736330753 | 16.556 |
| 26 | 66389631 | 7.822 | 56 | 71028895618181759 | 16.853 |
| 27 | 132844159 | 8.123 | 57 | 142904633121912833 | 17.158 |
| 28 | 266734335 | 8.426 | 58 | 285803715209210623 | 17.457 |
| 29 | 513010785 | 8.710 | 59 | 575488792479997953 | 17.761 |
| 30 | 1070075391 | 9.029 | 60 | 1148272730287255039 | 18.060 |
| 31 | 2055054945 | 9.313 | 61 | 2210621488441664865 | 18.345 |
| 32 | 4118168929 | 9.615 | 62 | 4572655407598512255 | 18.660 |
| 33 | 8268764385 | 9.917 | 63 | 9145099114469304447 | 18.961 |
| 34 | 14385384615 | 10.158 | 64 | 18397371556871432703 | 19.265 |
| 35 | 33875260545 | 10.530 | 65 | 36368285000677545089 | 19.561 |

disadvantage is more than offset by the many advantages which we have enumerated in this paper.

## ACKNOWLEDGMENT

The author gratefully acknowledges fruitful discussions with Professor Siavash Shahshahani and also the programming help of Mr. Bijan Aaraam for the preparation of this paper.

## REFERENCES

1. Metze, G. and J. E. Robertson, "Elimination of Carry Propagation in Digital Computers," *Proc. of the International Conf. on Information Processing*, pp. 389-396, Paris, June 1959.
2. Avižienis, A., "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers*, Vol. EC-10, No. 3, pp. 389-400, September 1961.
3. Svobada, A., "Rational Numerical System of Residue Classes," *Storje Na Zpracovani Informaci*, pp. 9-37, Sbornik V, Nakl. CSAV, Praha, 1957.
4. Svoboda, A., "The Numerical System of Residual Classes in Mathematical Machines," *Proceedings of International Conference on Information Processing*, pp. 419-422, UNESCO, Paris, June 1959, Butterworths, London, 1960.
5. Garner, H. L., "The Residue Number System," *IRE Transactions on Electronic Computers*, Vol. EC-8, No. 8, pp. 140-147, June 1959.
6. Avižienis, A., "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design,"

*IEEE Transactions on Computers,* Vol. C-20, No. 11, pp. 1322-1331, November 1971.

7. Avižienis, A., "Arithmetic Algorithms for Error-Coded Operands," *IEEE Transactions on Computers,* Vol. C-22, No. 6, pp. 567-572, June 1973.
8. Wallace, C. S., "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers,* Vol. EC-13, No. 1, pp. 14-17, February 1964.

## APPENDIX

## THEOREMS AND THEIR PROOFS

*Theorem 1:* $p_i = 2^{b_i} - 1$ and $p_j = 2^{b_j} - 1$ are relatively prime if and only if $b_i$ and $b_j$ are relatively prime.

*Proof:* (Only if part)—Let $b_i = zx$ and $b_j = zy$ with $z > 1$. Then, since $a^n - 1$ is divisible by $a - 1$, $p_i$ and $p_j$ are both divisible by $2^z - 1$ and, hence, they are not relatively prime.

(If part)—Suppose there exist pairs of integers of the form $2^{b_i} - 1$ and $2^{b_j} - 1$ which are not relatively prime while $b_i$ and $b_j$ are. Let $2^x - 1$ and $2^y - 1$ be one such pair with $x > y$ and $x + y$ a minimum among all such pairs. Let the odd prime number z divide $2^x - 1$ and $2^y - 1$. Then, z must also divide their difference

$$2^x - 2^y = 2^y(2^{x-y} - 1). \tag{17}$$

Since z cannot divide $2^y$, it must divide $2^{x-y} - 1$. But now z divides $2^{x-y} - 1$ and $2^y - 1$ with $x - y$ and y relatively prime (since, by assumption, x and y are relatively prime) and $(x - y) + y$ smaller than $x + y$ which was assumed to be a minimum among all such pairs; clearly a contradiction.

*Theorem 2:* If $b_i \geq 2$ for all i and if for $i \neq j$, we have $b_i \neq b_j$, then

$$\eta = \prod_{i=1}^{k} (1 - 2^{-b_i}) > 1 - 2^{-(\min_{i \leq k}(b_i) - 1)} \geq 1/2 \tag{18}$$

*Proof:* The second inequality is obvious upon noting that $\min_i (b_i) \geq 2$. To prove the first inequality, we first show, by induction on k, that:

$$\prod_{i=1}^{k} (1 - 2^{-b_i}) \geq 1 - \sum_{i=1}^{k} 2^{-b_i} \tag{19}$$

Clearly this is true for $k = 1$. To show that if the inequality holds for k it will also hold for $k + 1$, we multiply both sides of (19) by the positive value $(1 - 2^{-b_{k+1}})$ to get:

$$\prod_{i=1}^{k+1} (1 - 2^{-b_i}) \geq 1 - \sum_{i=1}^{k+1} 2^{-b_i} + b_{k+1} \cdot \sum_{i=1}^{k} 2^{-b_i} \tag{20}$$

The right-hand-side of (20) is clearly greater than the right-hand-side of (19) with k replaced by $k + 1$. Next, denoting $\min_{j \leq K}(b_i)$ by m, we write:

$$1 - \sum_{i=1}^{k} 2^{-b_i} > 1 - \sum_{x=m}^{\infty} 2^{-x} = 1 - 2^{-(m-1)} \tag{21}$$

Combining (21) with (19), we get the desired result.

*Theorem 3:* Given $b_1 < b_2 < \ldots < b_k$ and $b'_1 < b'_2 < \ldots < b'_k$ with $b_1 > b'_1$

and

$$\sum_{i=1}^{k} b_i = \sum_{i=1}^{k} b'_i = B, \tag{22}$$

we have

$$\prod_{i=1}^{k} (2^{b_i} - 1) > \prod_{i=1}^{k} (2^{b'_i} - 1). \tag{23}$$

*Proof:* Using Theorem 2 and the fact that $b'_1 \leq b_1 - 1$, we can write:

$$\begin{aligned}\prod_{i=1}^{k} (2^{b_i} - 1) &= 2^B \prod_{i=1}^{k} (1 - 2^{-b_i}) \\ &> 2^B (1 - 2^{-(b_1 - 1)}) \\ &\geq 2^B (1 - 2^{-b'_1})\end{aligned} \tag{24}$$

On the other hand:

$$\prod_{i=1}^{k} (2^{b'_i} - 1) = 2^B \prod_{i=1}^{k} (1 - 2^{-b'_i}) \leq 2^B (1 - 2^{-b'_1}) \tag{25}$$

Combining (24) and (25), we get the desired result.