Analysis of Tabular Methods for Modular Reduction

Behrooz Parhami

Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA

Abstract

Alternate table lookup schemes for the operation z mod p are presented and compared. The cases of a single table and two tables, including some new schemes, are analyzed completely and closed-form solutions for optimal designs are derived. Extension of the analysis and design techniques to more elaborate decomposition schemes are also discussed briefly. A novel multi-operand modular addition scheme is presented that can be used in implementing some multi-table schemes.

Keywords: Computer arithmetic, Modular arithmetic, Remainder finding, Residue number system, Table lookup.

1. Introduction

Table lookup is an attractive method for function evaluation in VLSI signal processing applications since it leads to the replacement of irregular area-intensive random-logic structures with much denser memory arrays. It is particularly useful for computing unary operations, although binary operations can also be handled in the case of short operands or where limited pre- and post-processing steps can be used to convert the binary operation into a unary one (examples are found in logarithmic number system addition and standard radix-2 multiplication through squaring) or to a set of smaller problems.

Modular reduction, or computing the residue of an integer z modulo a relatively small constant p, arises in many signal processing and general computing applications, particularly those involving residue number system (RNS) representations. Numerous examples of such applications have been documented in the literature [ALIA91], [HUNG94], [HUNG94b], [PARH93], [SODE86]. There are also instances of z mod p computation for large values of p, e.g., in cryptography [HUNG94a]. Even though we do not focus on such applications in this paper, some of our techniques are applicable to problem transformation and/or reduction in such cases.

Let $0 < z < m$. Thus $m - 1$ is the maximum possible value for the unsigned integer z. The binary representation of z has $b = \lceil \log_2 m \rceil$ bits. Residues mod p are represented as d-bit binary numbers, where $d = \lceil \log_2 p \rceil$.

We divide our discussion of tabular methods for computing z mod p into single-table and multiple-table methods. The direct single-table approach is only applicable to very short operands. We propose a modification to this approach that reduces the table size by a factor of at least $p/2$ at the cost of performing two additions after the lookup (Section 2). This renders the approach practical for somewhat wider operands and forms a basis against which improvements provided by multiple-table approaches can be judged.

We next turn to two-table methods as important special cases of the multiple-table approach, both because of their practicality of implementation and tractability of the associated analyses. The standard divide-and-conquer method, whereby z is divided into two parts, with each part used to address a separate table, is fully analyzed and a solution is obtained for the optimal split (Section 3).

We also present and analyze a novel two-table approach in which several high-order bits of z are used to determine a negative multiple of p that, if added to z, yields a result z' in the range $0<z'<m'$ with $p<m'<m$, z mod $p = z'$ mod p. The problem is then reduced to the tabular computation of z' mod p which is a less complex one (Section 4). We demonstrate, through an example, some tradeoffs involved in using one or the other two-table scheme.

More than two tables can be used to compute the d-bit residues $2^j z_{i:j}$ mod p, where $z_{i:j}$, $i>j$, is the binary number formed by bits i down to j of z. The z mod p operation can then be completed by modular multi-operand addition of d-bit integers. The two main parts of the problem are: (1) Devising a partitioning strategy, and (2) Performing modular multi-operand addition. Clearly solution of the first problem depends on the relative costs and delays of multiple-operand addition with varying number of operands as well as on the relationship between table size and complexity (area). Our analysis of the general multi-table scheme is as yet incomplete, and will be reported in future. However, a novel design for multi-operand modular addition, based on carry-save adders and table lookup, is presented that appears to be promising for the efficient implementation of multi-table schemes.

1058-6393/95 $4.00 © 1995 IEEE

2. Using a Single Table

2.1. Direct Table Lookup

The simplest way (conceptually) for computing $z \bmod p$ for $0 \le z < m$ is to construct a lookup table of size m which is indexed by z and from which the d-bit residue is obtained directly (Figure 1). The table size, in bits, for this direct approach is

$$T_1 = dm \qquad (1)$$

Clearly, this is impractical for all but very small values of m. We will use the table size given by Equation (1) as the basis against which all table size reduction and optimization schemes will be compared.

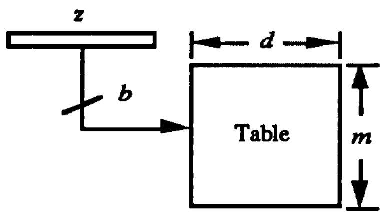

Figure 1. Simple lookup table for modular reduction.

2.2. Reducing the Table Size

Consider the identity:

$$z = 2^g \lfloor z/2^g \rfloor + z \bmod 2^g = 2^g z_{b-1:g} + z_{g-1:0} \qquad (2)$$

Reducing both sides of Equation (2) modulo p, we have

$$z \bmod p = [2^g z_{b-1:g} \bmod p + z_{g-1:0} \bmod p] \bmod p \qquad (3)$$

Let us take $g = d - 1$ so as to eliminate the second modulo computation. Noting that $z_{d-2:0} \bmod p = z_{d-2:0}$, we can rewrite (3) as :

$$z \bmod p = [2^{d-1} z_{b-1:d-1} \bmod p + z_{d-2:0}] \bmod p \qquad (4)$$

Assuming that the second modulo computation in Equation (4) is carried out by a trial subtraction (to determine if the value within the brackets is greater than or equal to p) and selection, a single table will be required to carry out the entire computation.

The required hardware architecture for this scheme is shown in Figure 2. The most significant $b - d + 1$ bits of z are used to address a u-word table (with $u = \lceil m/2^{d-1} \rceil$) which yields the needed d-bit residue $2^{d-1} z_{b-1:d-1} \bmod p$. This d-bit value is added to the least significant d–1 bits of z to produce a (d+1)-bit result representing $2^{d-1} z_{b-1:d-1} \bmod p + z_{d-2:0}$. We subtract p from this result and use the sign of the difference to control a d-bit two-way multiplexer whose inputs are the original value and the computed difference. The table size in this case is:

$$T_2 = du = d \lceil m/2^{d-1} \rceil \qquad (5)$$

The table size can thus be reduced by a factor of approximately 2^{d-1}, or at least $p/2$, compared to (1) at the cost of two adders and a multiplexer.

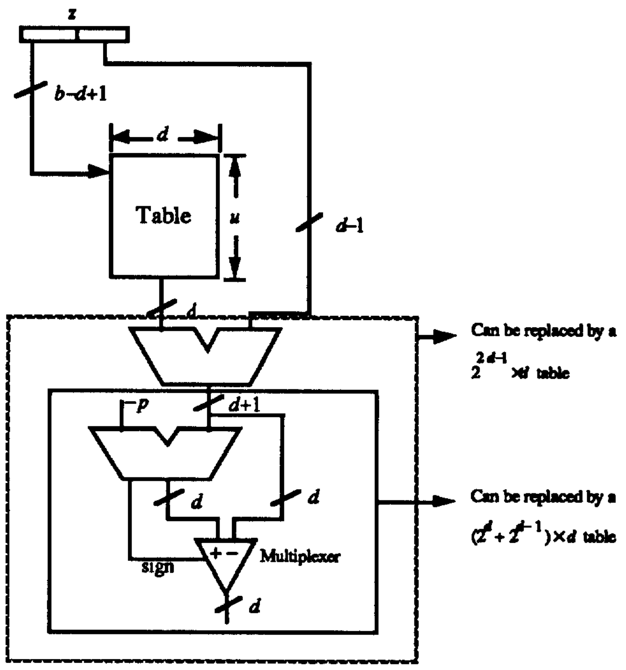
z
b–d+1
d
Table
u
d–1
d
Can be replaced by a $2^{2d-1} \times d$ table
–p
d+1
d
d
sign
+ –
Multiplexer
d
Can be replaced by a $(2^d + 2^{d-1}) \times d$ table

Figure 2. Single-table scheme with a smaller table.

Example 1: $p = 13$, $m = 2^{16}$, $d = 4$, $b = 16$

$T_1 = 4 \times 2^{16} = 2^{18}$ bits

$T_2 = 4 \times \lfloor 2^{16}/2^3 \rfloor = 2^{15}$ bits

so a factor of 8 in size reduction is achieved by using a 4-bit and a 5-bit adder plus a 4-bit multiplexer. ■

Example 2: $p = 13$, $m = 256$, $d = 4$, $b = 8$

$T_1 = 4 \times 256 = 2^{10}$ bits

$T_2 = 4 \times \lfloor 256/2^3 \rfloor = 2^7$ bits

The original table was small enough so that the savings are perhaps not worth the extra hardware in this case. ■

3. Using Two Tables

3.1. A Divide-and-Conquer Method

For $g \ge d$, Equation (2) can be the basis for the following two-table method. The least significant g bits of z, viz $z_{g-1:0}$, index a v-word table ($v = 2^g$) to obtain a d-bit residue. The most significant $b - g$ bits, $z_{b-1:g}$, index a second table with $v' = \lceil m/2^g \rceil$ words to obtain another d-bit residue. Adding of these residues and obtaining the final d-bit residue is done in a manner similar to that discussed in Section 2, just before Example 1. The above discussion leads to the table size:

$$T_3 = d(v + v') = d\,(2^g + \lceil m/2^g \rceil) \qquad (6)$$

3.2. The Optimal Split

Theorem 1: The table size T_3 is minimized if we choose

$$g = \lfloor \lceil \log_2 m \rceil / 2 \rfloor = \lfloor b/2 \rfloor \qquad (7)$$

Proof: The result is trivial for $m = 2^b$. We must show that it is valid for any m in the range $2^{b-1} < m < 2^b$. To show this, we prove that $T_3(g+1) - T_3(g) \le 0$ for $g < \lfloor b/2 \rfloor$ and $T_3(g+1) - T_3(g) \ge 0$ for $g \ge \lfloor b/2 \rfloor$.

$$T_3(g+1) - T_3(g) = d(2^{g+1} + \lceil m/2^{g+1} \rceil) - d(2^g + \lceil m/2^g \rceil)$$
$$= d(2^g + \lceil m/2^{g+1} \rceil - \lceil m/2^g \rceil) \quad (8)$$

For $g < \lfloor b/2 \rfloor$, we use the inequality $\lceil x/2 \rceil \ge \lceil x \rceil/2$ to rewrite Equation (8) as:

$$T_3(g+1) - T_3(g) = d(2^g + \lceil m/2^{g+1} \rceil - \lceil m/2^g \rceil)$$
$$\le d(2^g + \lceil m/2^g \rceil/2 - \lceil m/2^g \rceil) = d(2^g - \lceil m/2^g \rceil/2)$$
$$\le d(2^{\lfloor b/2 \rfloor - 1} - \lceil 2^{b-1}/2^{\lfloor b/2 \rfloor - 1} \rceil)$$
$$= d(2^{\lfloor b/2 \rfloor - 1} - 2^{\lceil b/2 \rceil - 1}) \le 0$$

The next to the last step above is justified by substituting the maximum value for g (i.e., $\lfloor b/2 \rfloor - 1$) and a lower bound for m (i.e., 2^{b-1}).

For $g \ge \lfloor b/2 \rfloor$, we use the inequality $\lceil 2x \rceil \le 2\lceil x \rceil$ to rewrite Equation (8) as:

$$T_3(g+1) - T_3(g) = d(2^g + \lceil m/2^{g+1} \rceil - \lceil 2 \times m/2^{g+1} \rceil)$$
$$\ge d(2^g + \lceil m/2^{g+1} \rceil - 2\lceil m/2^{g+1} \rceil) = d(2^g - \lceil m/2^{g+1} \rceil)$$
$$\ge d(2^{\lfloor b/2 \rfloor} - \lceil 2^b/2^{\lfloor b/2 \rfloor + 1} \rceil) = d(2^{\lfloor b/2 \rfloor} - 2^{\lceil b/2 \rceil - 1}) \ge 0$$

Again, the next to the last step above is justified by substituting the minimum value for g (i.e., $\lfloor b/2 \rfloor$) and an upper bound for m (i.e., 2^b). This concludes the proof. ■

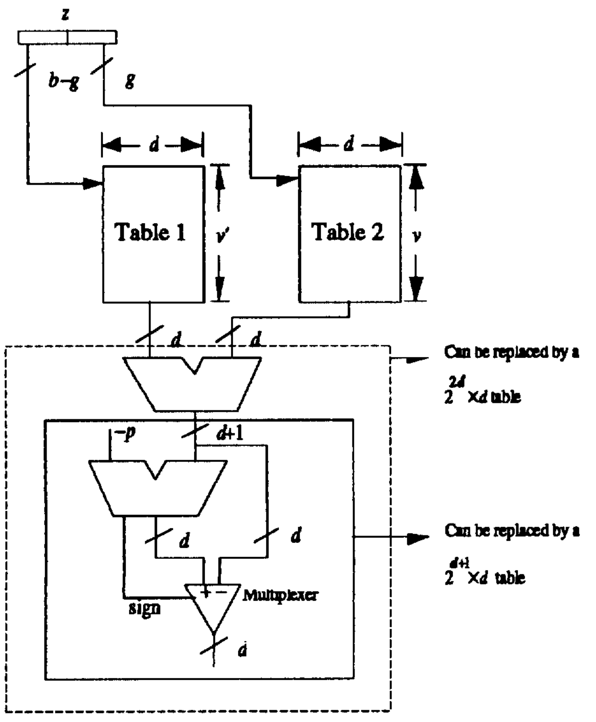
z
b−g
g
d
d
Table 1
v'
Table 2
v
d
d
Can be replaced by a $2^{2d} \times d$ table
−p
d+1
d
d
Multiplexer
sign
d
Can be replaced by a $2^{d+1} \times d$ table

Figure 3. Two-table modular reduction scheme based on divide-and-conquer strategy.

By Theorem 1, the minimal table size in this case is :

$$T_3^{\min} = d(2^{\lfloor b/2 \rfloor} + \lfloor m/2^{\lfloor b/2 \rfloor} \rfloor) \quad (9)$$

Example 3: $p = 13, m = 2^{16}, d = 4, b = 16$

$T_3^{\min} = 4(2^8 + 2^{16}/2^8) = 2^{11}$ bits

Comparing the above to 2^{18} bits or 2^{15} bits derived in Example 1, with one or two tables respectively, reveals the extent of savings. ■

Example 4: $p = 13, m = 256, d = 4, b = 8$

$T_3^{\min} = 4 \times (2^4 + 2^8/2^4)) = 2^7$ bits

The above result should be contrasted to 2^{10} bits obtained form Equation (1) and 2^7 bits obtained in Example 2. In general, the savings are less significant when b is not much larger then d. ■

4. A Novel Two-Table Scheme

4.1. The Basic Idea

The approach presented in this section consists of two phases. In Phase 1, several high-order bits of z ($0 \le z < m$) are used to determine what negative multiple of p should be added to z to yield a result z' in the range $0 \le z' < m'$ with $p < m' < m$ and $z \bmod p = z' \bmod p$. In phase 2, the simpler computation $z' \bmod p$ is performed.

Assuming $d' = \lceil \log_2 m' \rceil$, we have the two-table scheme depicted in Figure 4. The most significant $b - h$ bits of z, viz $z_{b-1:h}$, are used to access a w-word table ($w = \lceil m/2^h \rceil$) to obtain a d'-bit value. This value is the least significant d' bits of a negative multiple of p such that when it is added to z, the result z' is guaranteed to satisfy $0 \le z' \le m'$. A second m'-word table is used to obtain the d-bit final result $z' \bmod p$. The total table size required is thus:

$$T_4 = d'w + dm' = d'\lceil m/2^h \rceil + dm' \quad (10)$$

In the special case of $m' < 2p$, this second table can be eliminated and replaced by a subtractor and a multiplexer if desired, thus leading to a single-table scheme.

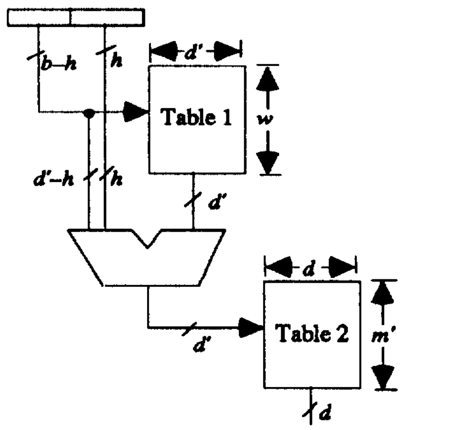

Figure 4. Two-table scheme based on successive refinement.

4.2. The Optimal Split

The key parameter here is m' which determines the number h of bits that can be ignored in Phase 1.

Theorem 2: The number h of bits that can be ignored in Phase 1 of the above scheme is at least $\lfloor \log_2(m'-p) \rfloor$.

Proof: Consider the interval $[0, m)$ divided into subintervals of width 2^h. In each subinterval, a certain negative multiple of p is added to z and the result must be in the interval $[0, m')$. consider the kth subinterval of width 2^h, where $0 \le k \le \lfloor m/2^h \rfloor$. The values of z in this subinterval range from $k \times 2^h$ to $(k+1) \times 2^h - 1$. Let the negative multiple of p to be added be $-M_k$ for the kth subinterval. Then, for each k, we must have:

$$k \times 2^h - M_k \times p \ge 0 \qquad (11a)$$

$$(k+1) \times 2^h - 1 - M_k \times p < m' \qquad (11b)$$

Conditions (11a) and (11b) restrict M_k in the range:

$$[(k+1) \times 2^h - m']/p \le M_k \le [k \times 2^h]/p \qquad (12)$$

A sufficient condition for (12) to yield an integer solution is for the right-hand side and left-hand side to differ by 1 unit or more. Hence the sufficient condition becomes

$$m' - 2^h \ge p \qquad (13)$$

which is equivalent to $h \le \lfloor \log_2(m'-p) \rfloor$. Since the sufficient condition given by Inequality (13) is not always necessary, in some cases it may be possible to take h to be larger than $\lfloor \log_2(m'-p) \rfloor$. ■

Using the result of Theorem 2, the table size T_4 given by Equation (10) can be rewritten as:

$$T_4 = d' \lceil m/2^{\lfloor \log 2(m'-p) \rfloor} \rceil + dm' \qquad (14)$$

The stage is now set for determining the optimal value of the key parameter m'.

Theorem 3: The table size T_4 given by Equation (14) is minimized if d' is selected so as to minimize the function

$$f(d') = d' \lceil m/2^{d'-1} \rceil + d \times 2^{d'-1} \qquad (15)$$

and the parameter m' is chosen to be:

$$m' = 2^{d'-1} + p \qquad (16)$$

Proof: Let $m' - p = x$. Then, since dp is a constant, to minimize T_4, the quantity

$$T_4 - dp = d' \lceil m/2^{\lfloor \log 2 x \rfloor} \rceil + dx \qquad (17)$$

must be minimized. For a given value of $\lfloor \log_2 x \rfloor$ in the denominator, dx is smallest when x is a power of 2; say, 2^y. Recall that $d' = \lceil \log_2 m' \rceil$. Therefore, we know that

$$2^{d'-1} < m' \le 2^{d'} \qquad (18)$$

Hence, from (18) we get $2^{d'-1} - p < x = 2^y \le 2^{d'} - p$, which leads to $y = d' - 1$ and $m' = 2^{d'-1} + p$, proving (16). The objective function to be minimized results from substituting $x = 2^{d'}-1$ in (17). ■

In the special case of $m = 2^b$, the function $f(d')$ to be minimized becomes:

$$f(d') = d' \times 2^{b-d'+1} + d \times 2^{d'-1} \qquad (19)$$

Differentiating (19) with respect to d' and equating with 0, we get:

$$d' = b/2 + 1 - (\log_2 d)/2 + (1/2)\log_2(d' - 1/\ln 2) \qquad (20)$$

Since $(1/2)\log_2(d' - 1/\ln 2)$ is small compared to d', Equation (20) can be solved by ignoring the last term on the right-hand side, and then adjusting the resulting solution to account for that term.

Example 5: $p = 13$, $m = 2^{16}$, $d = 4$, $b = 16$

From Equation (20), we get $d'^{\text{opt}} = 9$. This leads to $m' = 269$, $h = 8$, $w = 2^{b-h} = 2^8$, and the total table size:

$$T_4 = d'w + dm' = 9 \times 2^8 + 4 \times 269 = 3380 \text{ bits}$$

The above result should be contrasted to the results obtained in Examples 1 and 3. We note that the table size here is larger than that of Example 3 but that the simpler additional logic in Figure 4 compared to Figure 3 may make this approach more cost effective overall. ■

Example 6: To use the single-table version of our new scheme with the data as in Example 5, we must pick m' to be less than $2p$. Let us pick $m' = 21$ so that $h = 3$ bits of z can be ignored in the table lookup (Theorem 2). This results in $d' = 5$ and $w = 2^{15}$. Hence:

$$T_5 = d'w = 5 \times 2^{15} \text{ bits}$$

The resulting table is almost as large as the one needed for direct lookup and much larger than the one needed with the single-table approach of Subsection 2.2 (see Example 1). The fact that the peripheral logic needed here is somewhat simpler that than that of Figure 2, does not adequately compensate for the additional table size. ■

5. Using Multiple Tables

One can use more than two tables to compute the residues $2^j z_{i:j} \bmod p$ for various non-overlapping intervals $[i:j]$ such that they cover the entire index set $[b-1:0]$. The $z \bmod p$ operation can then be completed by modular multi-operand addition. Two questions immediately arise:

(1) What is a good partitioning strategy?

(2) How to perform modular multi-operand addition?

The above questions cannot be answered independently, as the optimal number and size of partitions is dependent on the cost function of the multi-operand addition scheme. There has been limited discussion of multi-operand modular addition in the literature [KOCC90], [PIES91], [PIES94] and even less discussion of its cost. Thus, at this point, the above questions cannot be answered in a reasonably general way.

We are working on new strategies for multi-operand modular addition that seem quite general and efficient. One scheme is based on a mix of carry-save adders and table lookup, as depicted in Figure 5. In this figure, solid lines represent d-bit numbers, unless otherwise specified, and dotted lines represent single bits. The idea in this example is to reduce six d-bit numbers to three d-bit numbers with the same modular sum. The 8-entry table contains the d-bit residues $k \times 2^d \bmod p$, for $0 \le k \le 3$.

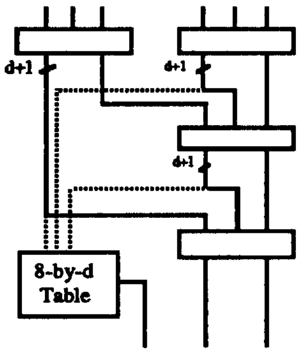

Figure 5. A strategy for multi-operand modular addition based on carry-save adders and table lookup.

Repeating such reductions, one can convert the multiple operands to a final set of 2 or 3 which can then be added and reduced by one of the methods discussed earlier.

Analysis of this modular reduction method and its effects on the initial partitioning will be reported in the near future. Different partitioning schemes yield several known methods of binary-to-residue conversion as special cases [ALIA84], [ALIA90], [ALIA93], [CAPO88], [PARH93a], [PARH94], [SHEN89].

6. Conclusion

Alternate table lookup schemes for performing the operation $z \bmod p$, where p is a relatively small constant, were presented and analyzed. Optimality results were obtained in the case of a single table and two different two-table schemes. Extension of these techniques to more elaborate decomposition schemes was briefly discussed.

References

[ALIA84] Alia, G. and E. Martinelli, "A VLSI Algorithm for Direct and Reverse Conversion from Weighted Binary Number System to Residue Number System", *IEEE Transactions on Circuit and Systems*, Vol. CAS-31, pp. 1033-1039, 1984.

[ALIA90] Alia, G. and E. Martinelli, "VLSI Binary-Residue Converters for Pipelined Processing", *The Computer Journal*, Vol. 33, pp. 473-474, 1990.

[ALIA91] Alia, G. and E. Martinelli, "A VLSI Modulo m Multiplier", *IEEE Transactions on Computers*, Vol. 40, No. 7, pp. 873-878, July 1991.

[ALIA93] Alia, G. and E. Martinelli, "On the Lower Bound to the VLSI Complexity of Number Conversion from Weighted to Residue Representation", *IEEE Transactions on Computers*, Vol. 42, No. 8, pp. 962-967, Aug. 1993.

[CAPO88] Capocelli, R.M. and R. Giancarlo, "Efficient VLSI Networks for Converting an Integer from Binary System to Residue Number System and Vice Versa", *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1425-1430, Nov. 1988.

[HUNG94] Hung, C.Y. and B. Parhami, "An Approximate Sign Detection Method for Residue Numbers and Its Application to RNS Division", *Computers & Mathematics with Applications*, Vol. 27, No. 4, pp. 23-35, 1994.

[HUNG94a] Hung, C.Y. and B. Parhami, "Fast RNS Division Algorithms for Fixed Divisors with Application to RSA Encryption", *Information Processing Letters*, Vol. 51, pp. 163-169, 1994.

[HUNG94b] Hung, C.Y. and B. Parhami, "Error Analysis of Approximate Chinese-Remainder-Theorem Decoding", *IEEE Transactions on Computers*, to appear.

[KOCC90] Koc, C.K. and C.Y. Hung, "Multi-Operand Modulo Addition Using Carry-Save Adders", *Electronic Letters*, Vol. 27, No. 6, pp. 361-363, 15 Mar. 1990.

[PARH93] Parhami, B. and H.-F. Lai, "Alternate Memory Compression Schemes for Modular Multiplication", *IEEE Transactions on Signal Processing*, Vol. 41, No. 3, pp. 1378-1385, Mar. 1993.

[PARH93a] Parhami, B., "Optimal Table-Lookup Schemes for Binary-to-Residue and Residue-to-Binary Conversions", *Proc. of the 27th Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 1993, Vol. 1, pp. 812-816.

[PARH94] Parhami, B. and C.Y. Hung, "Optimal Table Lookup Schemes for VLSI Implementation of Input/Output Conversions and Other Residue Number Operations", *Proc. of the IEEE Workshop on VLSI Signal Processing*, La Jolla, CA, Oct. 1994.

[PIES91] Piestrak, S.J., "Design of Residue Generators and Multi-Operand Modular Adders Using Carry-Save Adders", *Proc. of the Int'l Symp. on Computer Architecture*, Toronto, May 1991, pp. 100-107.

[PIES94] Piestrak, S.J., "Design of Residue Generators and Multioperand Modular Adders Using Carry-Save Adders", *IEEE Transactions on Computers*, Vol. 43, No. 1, pp. 68-77, Jan. 1994.

[SHEN89] Shenoy, A.P. and R. Kumaresan, "Residue to Binary Conversion for RNS Arithmetic Using Only Modular Look-Up Tables", *IEEE Transactions on Circuits and Systems*, Vol. 35, No. 9, pp. 1158-1162, Sep. 1989.

[SODE86] Soderstrand, M.A., W.K. Jenkins, G.A. Jullien, and F.J. Taylor (Editors), *Residue Number System Arithmetic*, IEEE Press, New York, 1986.