MAXIMAL REDUNDANCY SIGNED-DIGIT SYSTEMS

Tien Chi Chen*

The Chinese University of Hong Kong
Shatin, HONG KONG

ABSTRACT

The maximal redundancy signed-digit (MAXSD) number system has the highest redundancy within the carry-absorbing signed-digit number system proposed by Avizienis in 1961. The digital values for radix R lie in [1-R, R-1].

Its compatibility with both standard nonredundant systems and binary arithmetic makes it an excellent choice for multiprecision arithmetic on binary machines. The representations for finite numbers are however nonunique and can even be unbounded in wordlength; this is resolved by algorithms for partial or complete conversion to standard nonredundant notation without explicit carry propagation.

1. The Signed-Digit (SD) number system

A quarter century ago, A. Avizienis[2,3,4] studied digital number representations with signed digits under the following premises:

P1. Zero is represented by setting all digits to 0.

P2. The sign of the number is given by that of the lead nonzero digit;

P3. The negation of the number is obtained simply by reversing the sign of all nonzero digits;

P4. For addition the sum digit at a given position is completely determined by the inputs at both the same position and the position immediately to the right.

He observed that these requirements are satisfied by positional number systems with (implied) radix R > 2 characterized by the parameter q such that

$$-q \le d_k \le q; \text{ with } \tfrac{1}{2}(R+1) \le q \le R$$

Such systems not only obviate the need for a sign digit and provide immediate carry absorption in parallel addition/subtraction, but has the benefit that addition, subtraction, multiplication and division can all be carried out with the most significant digits emerging first, in the manner of on-line arithmetic (see, for instance Trevedi and Ercegovac[9]), not possible with nonredundant number systems.

Despite these advantages, SD schemes have not prevailed in the market. The built-in redundancy takes a toll in storage capacity, hardware cost and processing time. Moreover, the roundoff error is sensitive to detailed implementation.

A new opportunity arises with the recent strong interest in multiple precision arithmetic, where carry-propagation can be time-consuming. The emerging parallel computation designs, and vector arithmetic systems all favor a parallel processing approach to multiprecision arithmetic. These facts has motivated the present investigation.

2. Digital redundancy and recoding

An SD digit has (2q+1) distinct states, versus R in the standard nonredundant (NR) system with the same radix. The digital redundancy can be defined by

---

* Work performed at IBM Heidelberg Scientific Center, Tiergartenstrasse 15, D6900 Heidelberg, Federal Republic of Germany, Fall 1984.

$r \equiv 2q+1-R$
≥ 2 for $R \geq 3$.

During an addition, say between $A \equiv \{A_k\}$ and $B \equiv \{B_k\}$, the value $V \equiv (A_k + B_k) =$ is expressed as $cR+d$ and symbolized as the pair $[c,d]$, where d is an interim digit and c is a signed carry (called a transfer digit by Avizienis) which could be 0, +1, or -1. A nonzero carry is absorbed by the digit to its left usually; but if the latter possesses critical values q or -q respectively digital overflow occurs, calling for carry propagation.

A digital redundancy of 2 or greater in SD systems permits the (conceptual) recoding of the two critical values, to eject a nonzero carry ±1 beforehand, leaving an interim digit $d = \pm(q-R)$ which for $R>2$ is always subcritical, ready to absorb an arbitrary carry from the right. The recoding of other values from V to $[c,d] = [\pm 1, \pm(V-R)]$ are often possible, but does not serve the explicit purpose of carry absorption.

3. The MAXSD system

3.1. Characteristics

The highest redundancy within the SD context is reached with the MAXimal (redundancy) Signed-Digit (MAXSD) system, with $q = R-1$. It has the following features:

a. For given R, it has the largest capacity $(2R-1)$, and the greatest digital redundancy $r = (R-1)$.

b. It contains all other SD notations, as well as the standard nonredundant (NR) notation as special cases.

c. If $R = 2^p$, then the MAXSD digit is just a p-bit field.

d. Every nonzero digit can be recoded into a nontrivial carry-digit pair if needed. This curious fact has its uses as well as drawbacks.

3.2. MAXSD arithmetic

The MAXSD scheme allows simple hardware additions using digit-sized signed accumulators extended to contain the sum of two digits. The overflow bit together with the sign of the digit forms a signed carry. Normal operations, including the mandatory carry cases, are straightforward. The recoding is just taking the R's complement.

Localized additions, such as adding and subtracting 1's, and rounding in several different ways to bracket the final results, are well-served by the high-capacity MAXSD, often within a single digit.

Multiplications and divisions, however, would require a double-digit accumulator, or table look-up devices.

For $R=2^p$, the MAXSD digit is a p-bit field, and the recoding of q into R-q amounts to replacing a field of all-1's by all-0's with a trailing 1.

3.3. Addition with carry prediction

It is even possible to add in MAXSD using digit-size hardware, with neither extension nor overflow indicators. Even in such a case, the carries can be predicted with ease. The algorithm is, for every column:

a. If both input digits are zero, or if both digits are nonzero with opposing signs, normal addition can take place without recoding; the carry is 0.

b. If both addend digits are nonzero and equal in sign, recode either one of the digits. The subsequent addition follows (a) and will generate no extra nonzero carry;

c. If exactly one digit is 0, the the other digit already has the intermediate sum, and is recoded if necessary to avoid critical values.

Here the recoding in (b) is not confined to intermediate critical valus, but applies to nonzero digits of all types. Nonmaximal SD systems cannot do this, and would require costly scrutiny of the addend digits to achieve the carry prediction.

4. The wordlength excess problem

4.1. SD wordlength uncertainty

The SD numbers may possess more digits than their NR equivalents. The

MAXSD scheme aggravates the problem in having representations with unlimited wordlengths. This is seen from the following radix-10 example:

```
NR:        +(8 4)      (2 digits plus sign)
SD, q<8:   (1 -2 4)                (3 digits)
SD, q=8:   (8,4) = (1 -2 4)    (2-3 digits)
MAXSD:     (8 4) = (1 -2 4) = (1 -9 -2 4)
                  = (1 -9 -9 -2 4) =...
```

The leading digit 1 in MAXSD can be recoded as [1, (R-1)], which again has a leading digit 1, and can be recoded once more. Such a recoding of 1 is not explicitly practiced in MAXSD additions, but may result from multiple arithmetic operations.

This wordlength uncertainty is perhaps the greatest hidden problem in SD arithmetic. Clearly NR roundoff analysis cannot be applied directly without modification. Worse, one may find in integer arithmetic premature overflow indication, and in fractions, fictitious integer parts. In floating-point arithmetic a long apparent fraction length can cause loss of significant digits in some numbers, which during addition may force other addends to lose their significant digits as well.

EXAMPLE: The NR decimal fraction 0.84 may appear in SD as (1.-2 4), with an integer part, and in MAXSD as $(.1\ -9\ -9\ -9) \times 10^{+4} = 1.000$.

4.2. Wordlength excess in SD data

The SD wordlength problem is summarized below:

a. For a given number,

   The NR representation is unique and minimal in wordlength.

   Non-maximal SD schemes has nonunique representations, and possibly 2 distinct wordlengths.

   MAXSD systems has an unbounded variety of representations and wordlengths.

b. Nonzero wordlength excess over the NR notation occurs iff the SD digit is ±1, and the next nonzero digit (possibly separated by many 0's) has opposing sign.

c. For non-maximal SD systems,

   The excess never exceeds 1.

   The excess may be irreducible within the notation. For R = 10 and q = 7, (1 -2 4) cannot be reduced to (8 4).

   There are always values with wordlength ambiguity in representation. (5) = (1 -5) is a decimal example.

   With increasing q. the wordlength ambiguity increases, yet the number of irreducible cases decreases.

d. MAXSD representations for a given value allow arbitrarily large wordlength excesses. Fortunately all are reducible, if only because NR is a special case.

4.3. Reducing MAXSD wordlength excess

The MAXSD wordlength excesses are all easily reducible. For instance, the simple expedient of adding 0 to the number already rids the number of undesirable extra ±q's (q=R-1), bringing the wordlength down to the level of other SD number systems, with a excess of 1 at most.

To eliminate the excess altogether, one should be ready to transform the leading part of the number.

In the following we define

```
P = a positive definite digit
N = a negative definite digit
T = the digit (-1)
S = 1 if the number is positive
      definite
    0 if the number is 0
    T if the number is negative
      definite
```

Step 1. If the number is 0, all but one of the digits can be omitted. Else drop all leading 0 digits.

Step 2. The leading part is reducible iff it has the form

-S(TP...P), followed by -S(N or 0); or -S(TO...0P...P), followed by -S(N or 0)*

The reduction on the leading part consists of

a. Outside parentheses: S replces -S;
b. Inside: Replacing the lead T by 0;
c. Replacing the rest of the string by its R's complement;
d. Drop all leading 0 digits;

---

* The strings are actually S(1N,...,N), etc., rewritten with -S factor to show correspondence with R's complementation.

There will be no carries during the reduction.

EXAMPLES: (R = 10)

(0 0 0 0) --> (0)

(0 0 -3 1 0 -9)--> (-3 1 0 -9)

(0 0 1 -9 -9 -2 -4 8)
--> T(T 9 9 2 4), (8)
--> 1(0 0 0 7 6), (8) --> (7 6 8)

(0 0 T 0 0 9 9 2 4 -8)
--> 1(T 0 0 9 9 2 4), (-8)
--> T(0 9 9 0 0 7 6), (-8)
--> (-9 -9 0 0 -7 -6 -8)

Step 2 is strongly reminescent of the negation of a signed number in R's complement arithmetic, generalized here to signed digital values. The lead digit within the parentheses, namely T, behaves like a signed digit.

Step 2 can be economized by shortening the string of P values sampled:

In $S(T0\ldots0P_1,P_2,\ldots,P_m)$, only P_1 is truly needed;

In $S(TP_1, P_2,\ldots,P_m)$, P_1 alone suffices unless it equals (R-1); in which case one should continue sampling to include a second P value, if any.

EXAMPLES: (R = 10)

(0 0 1 -9 -9 -2 -4 8)
--> T(T 9 9 2), (-4 8)
--> 1(0 0 0 8), (-4 8) --> (8 -4 8)

(0 0 T 0 0 9 9 2 4 -8)
--> 1(T 0 0 9), (9 2 4 -8)
--> T(0 9 9 1), (9 2 4 -8)
--> (-9 -9 1 9 2 4 -8)

5. Conversion to NR notation

A fruitful approach to wordlength reduction in MAXSD would be to map directly into the NR notation, which has minimal wordlength. On the other hand, no conversion is needed to map from the standard NR number notations to enter MAXSD arithmetic.

The standard technique for back conversion to NR notation is to separate the parts with opposite signs into two words, and add them together in NR arithmetic, with the usual carry propagation.

An alternative technique for positive (negative) numbers would be to recode every negative (positive) digit but no others, add to zero in MAXSD arithmetic, repeat until all signs are equal. The number of iterations would be measurd by the longest string of zero digits terminating at a negative (positive) digit.

An intriguing question is, whether a carry-nonpropagating conversion mechanism to NR is possible. The answer is yes, in a manner of speaking, by applying the generalized R's complement technique to all strings containing digits of the wrong sign in parallel.

The digits in the given MAXSD word **W** is first subdivided into interlaced strings:

$$\mathbf{W} = \mathbf{V}, \mathbf{U}, \mathbf{V}, \mathbf{U}, \mathbf{V},\ldots, \mathbf{U}, \mathbf{V}$$

where **U** has the form $-S(N, \mathbf{M})$
$\mathbf{M} = (M_1, \ldots, M_m)$
M_k = P or 0, but never N;
also, M_m is never 0;

V has the form $-S(\mathbf{Q_0}, \mathbf{Q_1},\ldots\mathbf{Q_j})$
Q_k is never P, and some V's could be empty.

The recoding applies only to U (the string containing digits of sign Unlike that of the number itself):

$-S(N, M_1, \ldots, M_m) = -S(N, \mathbf{M})$
--> S((N+1), (R's compl. of **M**))

EXAMPLES: (R = 10)

```
    V U------- V-- U--- V-------
   (0 1 -9 0 0 -7 0 1 4 -2 3 9 0 )
-->(0 0  0 9 9  3 0 1 3  8 3 9 0 )

    V---------- U--------- U------ V
   (0 T -9 0 0 -7  0  1  4 -2  3  9 0 )
-->(0 T -9 0 0 -6 -9 -8 -6 -1 -6 -1 0 )
```

and the result will be in NR notation, yet no carry propagation ever explicitly occurs.

It may be said that the scanning effort to isloate the U-strings is nontrivial, and is similar to carry propagation. However, in scanning only 0 digits have assignment ambiguity calling for examination beyond neighboring digits. The interplay between MAXSD and NR numbers clearly bears further study in the future.

6. Multiprecision MAXSD arithmetic

As machines become faster and faster, the quality of the arithmetic results becomes an increasing concern. This has led to a strong interest in multiple-precision arithmetic; for a recent study using microprocessors see Aberth[1].

Kulisch and Miranker[7] has examined the use of a long accumulator of more than a thousands bits, covering the full range of products of all floating-point numbers, to accumulate results of vector dot products, for matrix computations to essentially full-length accuracy on otherwise short wordlength machines. Schemes using shorter accumulators exist, but are much less efficient. The schemes have been implemented in software[5] and hardware[6].

We have programmed multiprecision MAXSD arithmetic on the IBM Personal Computer APL system[8] in both floating-point and fixed-point, with a user-specifiable radix. At the largest allowed radix of 2^{26}, the limit on APL signed integers. APL provides no overflow indication, necessitating the carry prediction scheme of Section 3.3.

Multiplication and division are done by reverting to a half-length radix, which is no larger than 2^{28}. Additions in this lesser radix, also in MAXSD, but using effectively a double short-digit accumulator without carry prediction, was found to be much faster in APL for the same information content than that using the larger radix via carry prediction.

APL uses only one arithmetic engine, yet the processing of vectors and arrays outstrips serial digit-by-digit processing in speed. The experimental study could shed light on implementations using fast vector hardware, and highly overlapped machines, where a causal chain of events is handled far less efficiently as an equal number of otherwise unrelated ones. The carry non-propagation and the binary orientation of MAXSD should make a welcome difference.

7. Summary and conclusions

The signed-digit system, proposed a quarter-century ago, can benefit from re-examination in light of new need for multiprecision arithmetic and new vector machine hardware-software architecture.

The maximal signed digit system, compatible with standard NR systems on the one hand, and binary arithmetic on the other, is an excellent candidate. Problems of nonunique word lengths can be overcome in MAXSD, and the solution algorithm even offers the promise of NR arithmetic with no explicit carry propagation in the future.

ACKNOWLEDGMENT

The writer acknowledges stimulation received from Drs. William Carter and Ulrich Schauer from IBM, also Prof. Peter Linz from University of California at Davis.

REFERENCES

[1] O. Aberth, "Precise scientific computation with a microprocessor," IEEE Trans. Computers Vol. C-33, 685-690 (Aug. 1984).

[2] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," I. R. E. Trans. Elec. Computers Vol. EC-10, 389-400 (Sept. 1961).

[3] A. Avizienis, "On a flexible implementation of digital computer arithmetic," Proc. 1962 IFIP Congress, 664-668 (1962).

[4] A. Avizienis, "Binary-compatible signed-digit arithmetic, Proc. AFIPS Fall Joint Computer Conf. 1964, pp. 663-671.

[5] IBM SC33-6144-0, High-Accuracy Arithmetic Subroutine Library, Oct. 1983, 182 pages.

[6] IBM SA-22-7093-0, IBM System/370 RPQ, High Accuracy Arithmetic, Jan. 1984. 22 pages.

[7] U. W. Kulisch and W. L. Miranker, **Computer Arithmetic in Theory and Practice**, Academic Press (New York 1981). 249 pages. See Chapter 6.

[8] M. Tavera, M. Alfonseca, and J. Rojas, "The IBM personal computer APL system," ACM Conf. Proc. APL84, Helsinki, June, 1984, pp. 333-337.

[9] K. S. Trivedi and M. D. Ercegovac, "On-line algorithms for division and multiplication," IEEE Trans. Computers Vol. C-26, 681-687 (July 1977).