# Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations

BEHROOZ PARHAMI, SENIOR MEMBER, IEEE

*Abstract*—**Signed-digit (SD) number representation systems have been defined for any radix $r \geq 3$ with digit values ranging over the set $\{-\alpha, \cdots, -1, 0, 1, \cdots, \alpha\}$, where $\alpha$ is an arbitrary integer in the range $1/2\, r < \alpha < r$. Such number representation systems possess sufficient redundancy to allow for the annihilation of carry or borrow chains and hence result in fast propagation-free addition and subtraction. In this paper, we refer to the above as "ordinary" SD number systems and define generalized SD number systems which contain them as a special symmetric subclass. It is shown that the generalization not only provides a unified view of all redundant number systems which have proven useful in practice (including stored-carry and stored-borrowed systems), but also leads to new number systems not examined before. Examples of such new number systems are stored-carry-or-borrow systems, stored-double-carry systems, and certain redundant decimal representations.**

*Index Terms*—**Asymmetric signed-digit number systems, binary signed-digit numbers, computer arithmetic, number representation, redundant number systems, signed-digit arithmetic, stored-borrow representation, stored-carry representation.**

## I. Introduction

FOR any radix $r \geq 3$, there are one or more signed-digit (SD) number representation systems [2]–[4]. These *ordinary* SD (OSD) number systems correspond to different values of $\alpha$ in the range $1/2\, r < \alpha < r$, from the minimally redundant system ($\alpha = \lfloor 1/2\, r \rfloor + 1$) to the maximally redundant one ($\alpha = r - 1$), where $\alpha$ determines the set $\{-\alpha, \cdots, -1, 0, 1, \cdots, \alpha\}$ of the $2\alpha + 1$ digit values used. The most important property of OSD number representation systems is the possibility of performing carry-free addition and (by changing all the digit signs in the subtrahend) borrow-free subtraction.

The carry-free addition property of OSD number systems is best understood by a conceptual "recoding" process which replaces each $\pm\alpha$ value in the digit-by-digit *position sum* of two operands by $\pm(\alpha - r)$ and an outgoing transfer digit of $\pm 1$. The new digit value, which has a magnitude $r - \alpha$ in the range $0 < r - \alpha < 1/2\, r < \alpha$, always absorbs an incoming transfer digit of $\pm 1$, thus stopping its propagation.

Manuscript received May 15, 1987; revised December 19, 1987. This work was carried out while the author was a Visiting Professor at the University of Waterloo and was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants G1140, A3055, and A5515.

The author is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

IEEE Log Number 8930844.

Whenever long sequences of computations are to be performed on particular pieces of data, the one-time conversion and reconversion effort to/from the OSD representation is more than compensated for by the gain in computation speed. The crossover point is surpassed much more frequently in the case of maximally redundant OSD representations (for which $\alpha = r - 1$), since conventional radix-$r$ numbers can be interpreted as maximally redundant OSD numbers, with no need for the initial conversion [7]. However, maximally redundant OSD number systems may consume more storage space than the corresponding minimally redundant or intermediate OSD number systems.

The original definition of SD arithmetic uses a symmetric digit set and precludes the case of $r = 2$, since such a *binary signed-digit* (BSD) number system possesses insufficient redundancy for the general carry-free addition algorithm to be applicable. This is also the reason behind the requirement that $\alpha$ be greater than $1/2\, r$, even though $\alpha = 1/2\, r$ is a viable selection for a redundant number system if $r$ is even (see below). However, BSD numbers possess interesting properties [20] and have been in practical use for representing intermediate values in two's complement and high-speed multiplication schemes ever since multiplier recoding was introduced by Booth [6]. They have also been used in redundant quotient representation for the S–R–T division algorithm which was proposed independently by Sweeney [14], Robertson [22], and Tocher [24].

The generalization which is proposed here not only unifies the OSD and BSD number systems, but also covers as special cases all other useful redundant representations such as those offered by stored-carry and stored-borrow systems. It also provides valuable insight into the relationships of various redundant number representation systems and leads to some new representation methods which are interesting in their own right. The stored-carry-or-borrow representation system is probably the most important of these new methods and is thus explored in depth.

## II. Generalized Signed-Digit Number Systems

We define a *generalized signed-digit* (GSD) number system as a positional system with the digit set $\{-\alpha, -\alpha + 1, \cdots, \beta - 1, \beta\}$ with the conditions $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta + 1 > r$, where $r$ is the number representation radix. The excluded case of $\alpha + \beta + 1 = r$ results in nonredundant number representation systems which cover the conventional

0018-9340/90/0100-0089$01.00 

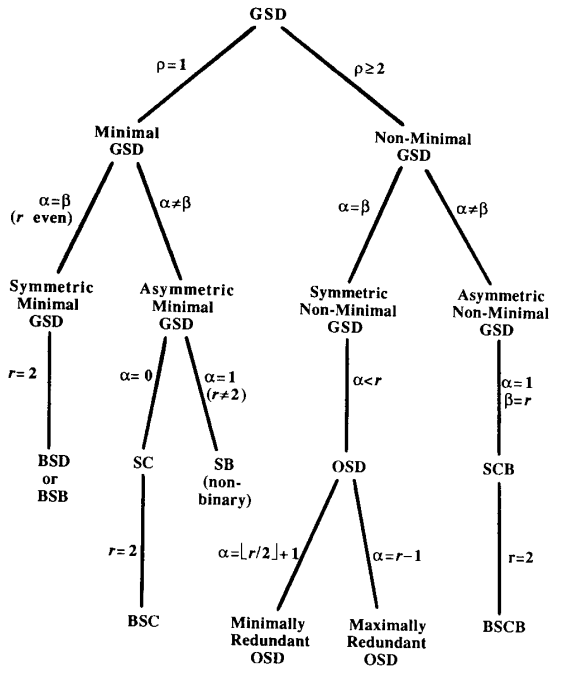



Fig. 1. The hierarchical relationships of redundant number representation systems.

radix-$r$ system with $\alpha = 0$ and $\beta = r - 1$ as a special case. The *redundancy index* of a GSD number system is defined as $\rho = \alpha + \beta + 1 - r$. GSD number systems cover the following systems as special cases [19]:

Binary stored-carry (BSC) number system: $r=2$, $\alpha=0$, $\beta=2$, $\rho=1$

Radix-$r$ stored-carry (SC) number systems: $\alpha=0$, $\beta=r$, $\rho=1$

BSD (also, binary stored-borrow or BSB) number system: $r=2$, $\alpha=\beta=1$, $\rho=1$

Radix-$r$ stored-borrow (SB) number systems: $\alpha=1$, $\beta=r-1$, $\rho=1$

Binary stored-carry-or-borrow (BSCB) number system: $r=2$, $\alpha=1$, $\beta=2$, $\rho=2$

Radix-$r$ stored-carry-or-borrow (SCB) number systems: $\alpha=1$, $\beta=r$, $\rho=2$

Minimally redundant symmetric SD number systems: $2\alpha=2\beta=r\geq 4$, $\rho=1$

OSD number systems: $r\geq 3$, $1/2\ r<\alpha=\beta<r$, $2\leq\rho<r$

Minimally redundant: $\alpha=\beta=\lfloor 1/2\ r\rfloor+1$, $2\leq\rho\leq 3$

Maximally redundant: $\alpha=\beta=r-1$, $\rho=r-1$.

Fig. 1 depicts the hierarchical relationships of these systems. The BSCB number system is equivalent to a redundant number representation system proposed for the design of a systolic binary counter [9], apparently without a realization that it was a signed-digit number system. Another systolic binary counter design [18] uses the BSCB number system in a different disguise.

It is interesting to note that GSD number systems actually do not cover the most general redundant representations. For example, the redundant number system with $r = 2$ and the digit set $\{\bar{3}, \bar{1}, 0, 2\}$ is not a GSD system. However, redundant systems with digit sets whose members are not necessarily consecutive integers do not appear to offer any advantage over GSD systems in terms of the speed or simplicity of arithmetic operations.

Most GSD number systems allow carry-free addition and borrow-free subtraction, just as OSD number systems do. This is not a trivial extension of the corresponding OSD property, since GSD number systems can have very strange digit sets (e.g., digits $\overline{16}$ to 15 with $r = 10$). A limited-carry addition algorithm is applicable to those GSD number systems which do not support carry-free addition.

## III. Carry-Free Addition of GSD Numbers

We start by considering the following carry-free addition algorithm which will be shown to be applicable to most GSD number representation systems.

*Algorithm 1 (carry-free addition):* Let the two numbers to be added have $x_i$ and $y_i$ as the $i$th digits. For each position $i$, a *position sum* $p_i = x_i + y_i$ is computed which is then broken into a *transfer digit* $t_{i+1}$ and an *interim sum* $w_i = p_i - rt_{i+1}$. The *final sum* digit is $s_i = w_i + t_i$ whose computation should produce no new transfer. ●

Let us see what is involved in computing the transfer digit $t_{i+1}$. From the digit set assumption ($-\alpha \leq x_i, y_i \leq \beta$) and the requirements of Algorithm 1, we have $-2\alpha \leq p_i \leq 2\beta$ and

$$-\alpha \leq w_i + t_i \leq \beta. \tag{1}$$

Substituting $p_i - rt_{i+1}$ for $w_i$ in (1), we get after some manipulation,

$$\frac{p_i-(\beta-t_i)}{r} \leq t_{i+1} \leq \frac{p_i+(\alpha+t_i)}{r}. \tag{2}$$

Let the range of transfer digits be

$$-\lambda \leq t_j \leq \mu \tag{3}$$

where $\lambda \leq \alpha$ and $\mu \leq \beta$ are nonnegative integers to be determined later. So, in the worst case, $t_{i+1}$ is in the range

$$\frac{p_i}{r} - \frac{\beta-\mu}{r} \leq t_{i+1} \leq \frac{p_i}{r} + \frac{\alpha-\lambda}{r}. \tag{4}$$

Fig. 2 depicts the range of values for $t_{i+1}$ as a function of $p_i$. A number of interesting observations can be made in connection with Fig. 2.

First, let us specialize Fig. 2 to the case of OSD numbers (Fig. 3). In general, there are overlap regions in Fig. 3 where two valid choices for $t_{i+1}$ exist. Maximum overlap occurs for maximally redundant OSD numbers ($\alpha = r - 1$). For such numbers, the transfer digit can sometimes be selected to be $\pm 2$ if $r \geq 4$. However, this only complicates the transfer digit selection process and the subsequent addition to compute $s_i = w_i + t_i$. Thus, it is advantageous to restrict the transfer digit values to the set $\{\bar{1}, 0, 1\}$ which is always adequate (see Fig. 3). The overlap amount is zero for even radices with $\alpha = 1/2\ r+1$ and is negative for odd radices with $\alpha = 1/2(r+1)$.

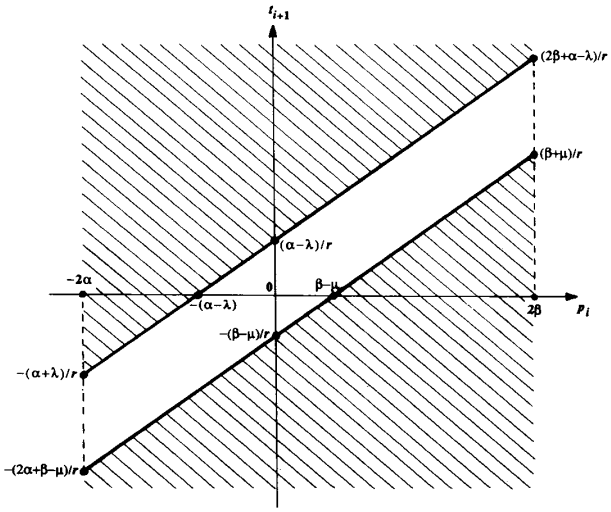



Fig. 2. The range of transfer digit values as a function of position sum for carry-free addition.

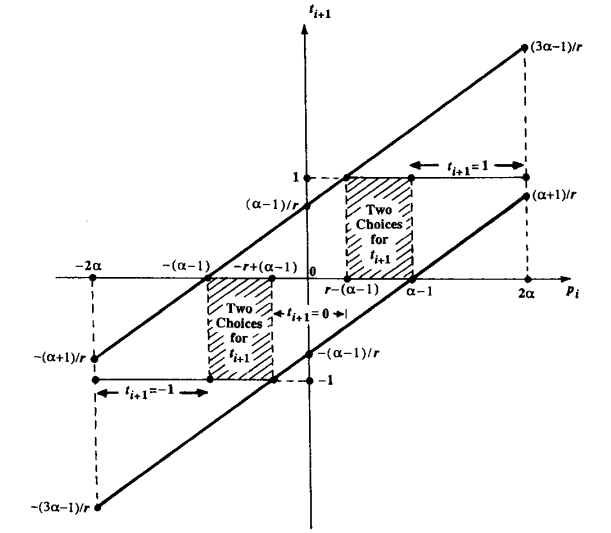



Fig. 3. The range of transfer digit values for ordinary signed-digit numbers.

No problem arises in the latter case due to the fact that an integer value for $p_i$ cannot fall in the "noncovered" region. Both of these cases correspond to minimally redundant OSD numbers.

The overlap region, if present and wide enough, can be used to simplify and speed up the adder design. As an example, consider the OSD number system with $r = 8$ and $\alpha = 7$. Here, the transfer digit may be 0 for $-6 \le p_i \le 6$, 1 for $p_i \ge 2$, and $\bar{1}$ for $p_i \le -2$. Therefore, it is sufficient to compute $p_i$ to an accuracy of $\pm 2$ and compare it to $\pm 4$ for selecting a proper transfer digit. Assuming sign-and-magnitude representation of the radix-8 digits, accuracy of $\pm 2$ can be obtained by ignoring the least significant bits of the two digits. The sum of the two high-order bits of the two digits must then be compared to $\pm 2$.

*Lemma 1:* A necessary and sufficient condition for the carry-free addition algorithm to be applicable is

$$\rho \ge \lambda + \mu. \tag{5}$$

*Proof:* The existence of an integer value for $t_{i+1}$ satisfying (4) implies

$$\frac{p_i}{r} - \frac{\beta-\mu}{r} \le \left\lfloor \frac{p_i}{r} + \frac{\alpha-\lambda}{r} \right\rfloor. \tag{6}$$

Let $p_i + \alpha - \lambda = r\gamma_i + \delta_i$, where $\delta_i$ $(0 \le \delta_i \le r-1)$ is the remainder of dividing $p_i + \alpha - \lambda$ by $r$. Then, (6) becomes

$$\frac{r\gamma_i + \delta_i - (\alpha-\lambda)}{r} - \frac{\beta-\mu}{r} \le \gamma_i.$$

Simplifying this inequality and noting that $\alpha + \beta = r + \rho - 1$ yields

$$\rho \ge \lambda + \mu - (r - 1 - \delta_i). \tag{7}$$

Inequality (7) must hold for all possible values of $\delta_i$. If we show that $\delta_i = r - 1$ for some $p_i$ in the range $-2\alpha \le p_i \le 2\beta$, the desired conclusion ($\rho \ge \lambda + \mu$) will be immediate. Consider the case $\delta_i = r - 1$ with $\gamma_i = 0$. We have in this case

$$p_i = \delta_i - (\alpha-\lambda) = r - 1 - (\alpha-\lambda). \tag{8}$$

Clearly, $p_i$ in (8) satisfies $p_i \ge -2\alpha$. It satisfies $p_i \le 2\beta$ if we have $\rho \ge \lambda - \beta$. This concludes the proof for the case $\rho \ge \lambda - \beta$. If $\rho < \lambda - \beta$, let $\delta_i = r - 1$ with $\gamma_i = -1$. We have in this case

$$p_i = -r + \delta_i - (\alpha-\lambda) = -1 - (\alpha-\lambda). \tag{9}$$

Clearly, $p_i$ in (9) satisfies $p_i \le 2\beta$. It satisfies $p_i \ge -2\alpha$ if we have $\alpha + \lambda \ge 1$. According to the assumption $\rho < \lambda - \beta$, we can write

$$\alpha + \lambda > \alpha + \rho + \beta = 2\rho + r - 1.$$

The conclusion $\alpha + \lambda \ge 1$ is immediate. ●

It is apparent from our previous discussion that the transfer digit selection process consists of a number of (approximate) comparisons between $p_i$ and known constants. Thus, the addition algorithm is simplified if the set of possible values for transfer digits is of minimal size.

*Lemma 2:* The set $\{-\lambda, -\lambda + 1, \cdots, \mu - 1, \mu\}$ of possible transfer digit values for carry-free addition of GSD numbers is of minimal size if we choose

$$\lambda^{\min} = \left\lceil \frac{\alpha}{r-1} \right\rceil \tag{10}$$

$$\mu^{\min} = \left\lceil \frac{\beta}{r-1} \right\rceil. \tag{11}$$

*Proof:* To determine the minimal value for $\lambda$, we set $p_i = -2\alpha$ in (4) to obtain

$$-\frac{2\alpha+\beta-\mu}{r} \le \min_{p_i}(t_{i+1}) \le -\frac{\alpha+\lambda}{r}.$$

To minimize the value of $\lambda$, we select

$$\max(-\lambda) = \max_{p_i}(\min(t_{i+1})) = \left\lfloor -\frac{\alpha+\lambda}{r} \right\rfloor$$

which in turn yields

$$\lambda^{\min} = \left\lceil \frac{\alpha+\lambda^{\min}}{r} \right\rceil . \tag{12}$$

To verify that (10) is indeed the solution of (12), we substitute and verify the following equality:

$$\left\lceil \frac{\alpha}{r-1} \right\rceil = \left\lceil \frac{1}{r}\left(\alpha + \left\lceil \frac{\alpha}{r-1} \right\rceil\right) \right\rceil . \tag{13}$$

Let $\alpha = (r-1)\gamma + \delta$, where $0 \le \delta < r-1$. Then, (13) becomes

$$\left\lceil \frac{\delta}{r-1} \right\rceil = \left\lceil \frac{1}{r}\left(\delta + \left\lceil \frac{\delta}{r-1} \right\rceil\right) \right\rceil . \tag{14}$$

Equality (14) clearly holds for $\delta = 0$. For $1 \le \delta < r-1$, we have $\lceil \delta/(r-1) \rceil = 1$ and (14) becomes

$$1 = \left\lceil \frac{\delta+1}{r} \right\rceil$$

which is cleary satisfied for all the values of $\delta$ in the assumed range. A similar argument shows that

$$\mu^{\min} = \left\lceil \frac{\beta+\mu^{\min}}{r} \right\rceil$$

which is satisfied by (11). ●

*Corollary 1:* In a GSD number system, the set of possible transfer digit values must have at least $\lceil \rho/(r-1) \rceil + 2$ members if the carry-free addition algorithm is to be applicable.

*Proof:* The set of possible transfer digit values must have at least $\lambda^{\min} + \mu^{\min} + 1$ members. We have from Lemma 2

$$\lambda^{\min} + \mu^{\min} + 1 = \left\lceil \frac{\alpha}{r-1} \right\rceil + \left\lceil \frac{\beta}{r-1} \right\rceil + 1$$
$$\ge \left\lceil \frac{\alpha+\beta}{r-1} \right\rceil + 1 = \left\lceil \frac{\rho}{r-1} \right\rceil + 2.$$

This concludes the proof. ●

An immediate implication of Corollary 1 is that at least three different values are needed for the transfer digit and that a simple binary carry (or borrow) is insufficient.

*Theorem 1:* For the carry-free addition defined by Algorithm 1 to be applicable to a GSD number system, it is necessary and sufficient to have $\rho \ge 3$ if either $\alpha$ or $\beta$ is equal to 1 and $\rho \ge 2$ otherwise, with $r > 2$ in both cases.

*Proof:* From Lemma 1, we must have $\rho \ge \lambda + \mu$. Thus, assuming minimal values for $\lambda$ and $\mu$, we can write

$$\rho \ge \lambda^{\min} + \mu^{\min} = \left\lceil \frac{\alpha}{r-1} \right\rceil + \left\lceil \frac{\beta}{r-1} \right\rceil \ge \left\lceil \frac{\alpha+\beta}{r-1} \right\rceil . \tag{15}$$

Noting that $\alpha + \beta = r - 1 + \rho$, we conclude that carry-free addition is possible only if

$$\rho \ge \left\lceil \frac{\alpha+\beta}{r-1} \right\rceil = 1 + \left\lceil \frac{\rho}{r-1} \right\rceil . \tag{16}$$

Inequality (16) cannot be satisfied for $r = 2$. Thus, Algorithm 1 is inapplicable to radix-2 GSD numbers, regardless of the choice of values for $\alpha$ and $\beta$. Similarly, a GSD number system with $\rho = 1$ (or $\alpha + \beta = r$) does not lend itself to carry-free addition and is in a sense insufficiently redundant for this purpose. For $r > 2$, necessity of $\rho \ge 2$ is implied by (16). To prove the sufficiency of $\rho \ge 2$ when $\alpha, \beta \ne 1$, we reconsider (15) and substitute $r - 1 + \rho - \alpha$ for $\beta$:

$$\rho \ge \left\lceil \frac{\alpha}{r-1} \right\rceil + \left\lceil \frac{\beta}{r-1} \right\rceil = \left\lceil \frac{\alpha}{r-1} \right\rceil + \left\lceil \frac{\rho-\alpha}{r-1} \right\rceil + 1. \tag{17}$$

For $\alpha = 0 \bmod(r-1)$, the sufficient condition given by (17) is identical to (16) and thus $\rho \ge 2$ is necessary and sufficient. Now let $\alpha = \delta \bmod(r-1)$; that is $\alpha = (r-1)\gamma + \delta$ for some $\gamma$, with $0 < \delta < r-1$. In this case, the sufficient conditions given by (17) can be reduced to

$$\rho \ge 2 + \left\lceil \frac{\rho-\delta}{r-1} \right\rceil . \tag{18}$$

For $\delta = 1$, inequality (18) is satisfied iff $\rho \ge 3$. For $2 \le \delta \le r-1$, inequality (18) is satisfied iff $\rho \ge 2$. The proof is complete upon noting that $\delta = 1$ and $\rho = 2$ imply either $\alpha = 1$ or $\alpha = r$ ($\beta = 1$). ●

*Algorithm 2 (transfer digit selection for carry-free addition):* The transfer digit $t_{i+1}$ is selected to be $k$ iff $C_k \le p_i < C_{k+1}$, where $C_{-\lambda} = -\infty$, $C_{\mu+1} = \infty$ and each $C_j$ ($-\lambda < j \le \mu$) is a known comparison constant to be specified later. ●

Clearly, the values of comparison constants in Algorithm 2 directly affect the complexity of the GSD adder. In general, there may be several valid choices for each $C_j$ and thus the one which results in the simplest possible hardware realization can be selected. The following theorem specifies the range of valid choices for $C_j$.

*Theorem 2:* For carry-free addition of GSD numbers, the comparison constants $C_k$ ($-\lambda < k \le \mu$) of Algorithm 2 must satisfy the following constraints:

$$kr - (\alpha - \lambda) \le C_k \le (k-1)r + \beta - \mu + 1. \tag{19}$$

*Proof:* From inequality (4), we see that the selection $t_{i+1} = k - 1$ is valid only if

$$p_i \le (k-1)r + \beta - \mu. \tag{20}$$

Similarly, the selection $t_{i+1} = k$ is valid only if

$$p_i \ge kr - (\alpha - \lambda). \tag{21}$$

So the boundary between choosing $k-1$ and choosing $k$ (i.e., $C_k$) must lie between $kr - (\alpha - \lambda)$ and $(k-1)r + \beta - \mu +$

1, provided that we have

$$kr-(\alpha-\lambda)\leq(k-1)r+\beta-\mu+1.$$

Considering that $\alpha + \beta - r = \rho - 1$, the above condition is equivalent to $\rho \geq \lambda + \mu$ which is the same as the condition of Lemma 1.

## IV. Limited-Carry Addition of GSD Numbers

Next, we consider a limited-carry addition algorithm which is applicable to all GSD number systems. This algorithm finds applications in cases where Algorithm 1 cannot be used. According to Theorem 1, this happens when we have one of the following situations: 1) $r = 2$, 2) $\rho = 1$, or 3) $\rho = 2$, with either $\alpha$ or $\beta$ equal to 1. Examples of GSD systems in these categories are the SC, SB, and SCB representation methods, including of course their radix-2 special cases.

*Algorithm 3 (limited-carry addition):* Let the two numbers to be added have $x_i$ and $y_i$ as the $i$th digits. In stage 1, for each position $i$, a *position sum* $p_i = x_i + y_i$ is computed and used to generate a *range estimate* $e_{i+1}$ for the final transfer digit $t_{i+1}$. In stage 2, the position sum $p_i$ and the range estimate $e_i$ are used to compute a *transfer digit* $t_{i+1}$ and an *interim sum* $w_i = p_i - rt_{i+1}$. The *final sum* digit is $s_i = w_i + t_i$ whose computation should produce no new transfer. ●

The new aspects of this algorithm with respect to Algorithm 1 are 1) computation of range estimates, and 2) computation of $t_{i+1}$ as a function of both $p_i$ and $e_i$. The range estimate $e_i$ may be presented in many different formats. In the simplest case, it is a binary indicator restricting $t_i$ to one of two subsets (not necessarily disjoint) of the set $\{-\lambda, -\lambda + 1, \cdots, \mu - 1, \mu\}$ of possible transfer digit values. In the most general case, the range estimate $e_i$ is $k$-valued and identifies one of $k$ (possibly overlapping) subintervals of the closed interval $[-\lambda, \mu]$ as containing $t_i$.

In our subsequent discussion, we consider the simplest case where a binary range estimate is used and show that this is sufficient for limited-carry addition in all cases. Let $\lambda'$ and $\mu'$ be constant integers satisfying

$$-\lambda<-\lambda'\leq\mu'<\mu. \tag{22}$$

Unlike $\lambda$ and $\mu$ which are assumed to be nonnegative, $\lambda'$ and $\mu'$ can also be negative. The minus sign of $-\lambda'$ in (22) is only used for convenience since it allows simple application of our previous results in this case. The binary range estimate $e_j \in \{l, h\}$ restricts the transfer digit $t_j$ into one of the two closed subintervals; the *low* subinterval $[-\lambda, \mu']$ and the *high* subinterval $[-\lambda', \mu]$. For example, if $\lambda' = \mu' = 0$, then the range estimate determines the sign of the transfer digit (with 0 considered both positive and negative).

Let us see what is involved in computing $e_{i+1}$ and $t_{i+1}$. Inequalities (1) and (2) remain valid here. The correspondence between range estimates and possible transfer digit values is as follows:

$$\text{if } e_i=l \text{ then } -\lambda\leq t_i\leq\mu' \tag{23a}$$

$$\text{if } e_i=h \text{ then } -\lambda'\leq t_i\leq\mu. \tag{23b}$$

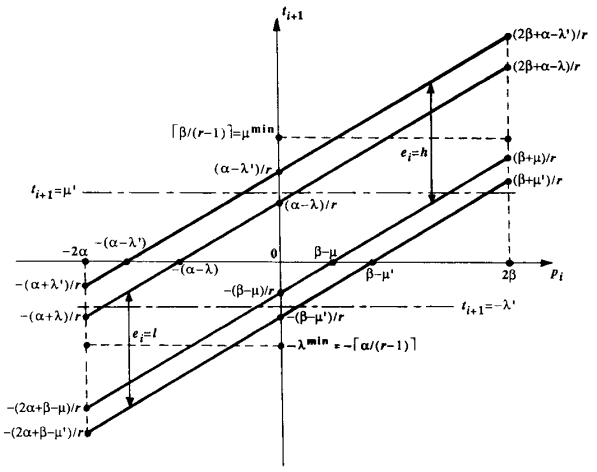



Fig. 4. The range of transfer digit values as a function of position sum for limited-carry addition.

Using inequalities (23), in the worst case, $t_{i+1}$ is in the range

$$\text{if } e_i=l \text{ then } \frac{p_i}{r}-\frac{\beta-\mu'}{r}\leq t_{i+1}\leq\frac{p_i}{r}+\frac{\alpha-\lambda}{r} \tag{24a}$$

$$\text{if } e_i=h \text{ then } \frac{p_i}{r}-\frac{\beta-\mu}{r}\leq t_{i+1}\leq\frac{p_i}{r}+\frac{\alpha-\lambda'}{r}. \tag{24b}$$

Fig. 4 depicts the range of $t_{i+1}$ as a function of $p_i$ and the incoming range estimate $e_i$.

Intuitively, the key to the success of the limited-carry addition algorithm in cases where the carry-free algorithm fails is the "expanded" range of $t_{i+1}$ which permits the selection of a value for $t_{i+1}$ when no acceptable value exists in the original nonexpanded range.

Obviously, we must make sure that the range estimate $e_{i+1}$ for the value of $t_{i+1}$ can be generated based on a simple test of $p_i$.

*Algorithm 4 (range estimate for limited-carry addition):* Select $e_{i+1} = l$ iff $p_i < E$ and $e_{i+1} = h$ iff $p_i \geq E$, where $E$ is a known comparison constant to be specified later. ●

*Theorem 3:* For limited-carry addition of GSD numbers, the comparison constant $E$ of Algorithm 4 must satisfy the following constraints:

$$-(\lambda'+1)r+\beta-\mu'<E\leq(\mu'+1)r-(\alpha-\lambda'). \tag{25}$$

*Proof:* Referring to Fig. 5, we note that for $p_i < F$ ($F$ being the value of $p_i$ at the intersection of the horizontal line $t_{i+1} = \mu' + 1$ with the uppermost oblique line), $t_{i+1}$ is at most equal to $\mu'$ regardless of the value of $e_i$. Thus, a range estimate $e_{i+1} = l$ can be generated for $p_i < F$. Similarly, for $p_i > D$ ($D$ being the value of $p_i$ at the intersection of the horizontal line $t_{i+1} = -\lambda' - 1$ with the lowermost oblique line), $t_{i+1}$ is at least equal to $\lambda'$ regardless of the value of $e_i$. Thus, a range estimate $e_{i+1} = h$ can be generated for $p_i > D$.

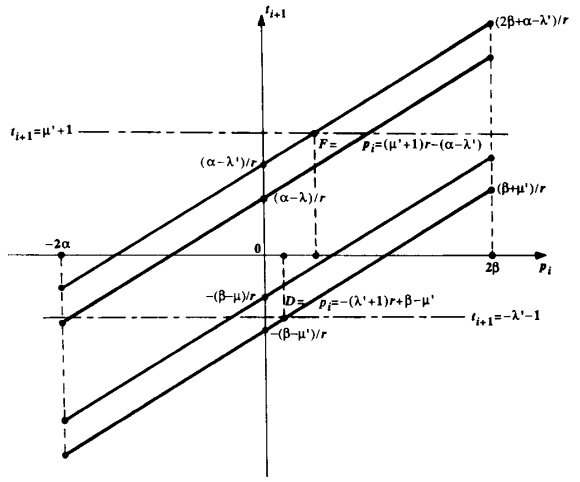



Fig. 5. Generating a binary range estimate $e_{i+1}$ for the transfer digit $t_{i+1}$.

Therefore, valid choices for $E$ are in the range $D < E \leq F$. Substituting the expressions for $D$ and $F$ from Fig. 5, we obtain the desired result. ●

*Algorithm 5 (transfer digit selection for limited-carry addition):* The transfer digit $t_{i+1}$ is selected to be $k$ iff $C_k(e_i) \leq p_i < C_{k+1}(e_i)$, where $C_{-\lambda}(l) = C_{-\lambda}(h) = -\infty$, $C_{\mu+1}(l) = C_{\mu+1}(h) = \infty$, and each $C_j(e_i)$ $(-\lambda < j \leq \mu, e_i \in \{l, h\})$ is a known comparison constant to be specified later. ●

Because of the need for taking $e_i$ into account, Algorithm 5 implies a more complex hardware implementation than its carry-free counterpart (Algorithm 2).

*Theorem 4:* For limited-carry addition of GSD numbers, the comparison constants $C_k(l)$ and $C_k(h)$ $(-\lambda < k \leq \mu)$ of Algorithm 5 must satisfy the following constraints:

$$kr-(\alpha-\lambda)\leq C_k(l)\leq(k-1)r+\beta-\mu'+1 \quad (26a)$$

$$kr-(\alpha-\lambda')\leq C_k(h)\leq(k-1)r+\beta-\mu+1. \quad (26b)$$

*Proof:* Similar to Theorem 2, with $\mu$ replaced by $\mu'$ in one pass to get the constraints on $C_k(l)$ and $\lambda$ replaced by $\lambda'$ in another pass for $C_k(h)$ conditions. ●

*Lemma 3:* Necessary and sufficient conditions for the limited-carry addition algorithm to be applicable to a GSD number system are

$$\rho\geq\max(\lambda+\mu',\ \lambda'+\mu) \quad (27)$$

$$\lambda'+\mu'\geq\left\lfloor\frac{\rho}{r+1}\right\rfloor. \quad (28)$$

*Proof:* The necessity of (27) is established similar to the proof of Lemma 1, with $\mu$ replaced by $\mu'$ in one pass (giving $\rho \geq \mu' + \lambda$) and $\lambda$ replaced by $\lambda'$ in another pass (giving $\rho \geq \mu + \lambda'$). The necessity of (28) results from the requirement $D < F$ in the proof of Theorem 3. Substituting the expressions for $D$ and $F$ from Fig. 5, we obtain

$$-(\lambda'+1)r+\beta-\mu'<(\mu'+1)r-(\alpha-\lambda').$$

Using the identity $\alpha + \beta = r - 1 + \rho$, the above condition becomes

$$\lambda'+\mu'>\frac{\rho}{r+1}-1. \quad (29)$$

Condition (29) is the same as (28) since $\lambda' + \mu'$ is an integer. ●

*Theorem 5:* The limited-carry addition defined by Algorithm 3 is applicable to all GSD number systems.

*Proof:* Substituting the values of $\lambda^{\min}$ and $\mu^{\min}$ in (27), we obtain

$$\rho\geq\max\left(\left\lceil\frac{\alpha}{r-1}\right\rceil+\mu',\ \lambda'+\left\lceil\frac{\beta}{r-1}\right\rceil\right). \quad (30)$$

Inequality (30) and the conditions $\lambda' < \lambda^{\min}$ and $\mu' < \mu^{\min}$ dictate

$$\lambda'\leq\min\left(\left\lceil\frac{\alpha}{r-1}\right\rceil-1,\ \rho-\left\lceil\frac{\beta}{r-1}\right\rceil\right) \quad (31a)$$

$$\mu'\leq\min\left(\left\lceil\frac{\beta}{r-1}\right\rceil-1,\ \rho-\left\lceil\frac{\alpha}{r-1}\right\rceil\right). \quad (31b)$$

We need only check that valid choices for $\lambda'$ and $\mu'$ satisfying (28) and (31) exist for cases where carry-free addition is impossible; namely for the cases 1) $r = 2$, 2) $\rho = 1$, 3) $\rho = 2$, with either $\alpha$ or $\beta$ equal to 1. For $r = 2$, conditions (28) and (31) become

$$\lambda'+\mu'\geq\left\lfloor\frac{\rho}{3}\right\rfloor \quad (32a)$$

$$\lambda'\leq\min(\alpha-1,\ \rho-\beta)=\alpha-1 \quad (32b)$$

$$\mu'\leq\min(\beta-1,\ \rho-\alpha)=\beta-1. \quad (32c)$$

It is sufficient to show that at least one pair of values for $\lambda'$ and $\mu'$ satisfy the conditions of (32). Let $\lambda' = \alpha - 1$ and $\mu' = \beta - 1$. Then, $\lambda' + \mu' = \alpha + \beta - 2 = \rho - 1$, which is clearly not less than $\lfloor\rho/3\rfloor$ since $\rho \geq 1$. This concludes the proof for $r = 2$. Next consider the simplified versions of (28) and (31) for $\rho = 1$:

$$\lambda'+\mu'\geq0 \quad (33a)$$

$$\lambda'\leq\min\left(\left\lceil\frac{\alpha}{r-1}\right\rceil-1,\ 1-\left\lceil\frac{r-\alpha}{r-1}\right\rceil\right) \quad (33b)$$

$$\mu'\leq\min\left(\left\lceil\frac{r-\alpha}{r-1}\right\rceil-1,\ 1-\left\lceil\frac{\alpha}{r-1}\right\rceil\right). \quad (33c)$$

Consider the three cases $\alpha = 0$, $\alpha = r$, and $0 < \alpha < r$. For $\alpha = 0$, the values $\lambda' = -1$ and $\mu' = 1$ satisfy the conditions given by (33). For $\alpha = r$, the values $\lambda' = 1$ and $\mu' = -1$ satisfy the conditions given by (33). Finally, for $0 < \alpha < r$, the values $\lambda' = \mu' = 0$ satisfy the required conditions. This concludes the proof for $\rho = 1$. For the final case of $\rho = 2$ with either $\alpha$ or $\beta$ equal to 1, we consider the two possibilities: $\alpha = 1$ and $\alpha = r$. The required conditions in this case are

$$\lambda' + \mu' \geq 0 \tag{34a}$$

$$\lambda' \leq \min\left( \left\lceil \frac{\alpha}{r-1} \right\rceil - 1,\ 2 - \left\lceil \frac{r+1-\alpha}{r-1} \right\rceil \right) \tag{34b}$$

$$\mu' \leq \min\left( \left\lceil \frac{r+1-\alpha}{r-1} \right\rceil - 1,\ 2 - \left\lceil \frac{\alpha}{r-1} \right\rceil \right). \tag{34c}$$

It is easily seen that in both cases of $\alpha = 1$ and $\alpha = r$, the values $\lambda' = \mu' = 0$ satisfy the conditions given by (34). ●

## V. Ordinary Signed-Digit Number Systems

Ordinary signed-digit (OSD) number representation systems have been defined for any radix $r \geq 3$ with digit values ranging over the set $\{-\alpha, \cdots, -1, 0, 1, \cdots, \alpha\}$, where $\alpha$ is an arbitrary integer in the range $1/2\, r < \alpha < r$. An OSD number system has a redundancy index in the range $2 \leq \rho < r$. The fact that $1 < 1/2\, r < \alpha < r$ implies that $\alpha \neq 1 \bmod (r - 1)$ and thus the carry-free addition algorithm is applicable even when $\rho = 2$. We have for all OSD number systems

$$\lambda^{\min} = \mu^{\min} = \left\lceil \frac{\alpha}{r-1} \right\rceil = 1.$$

For minimally redundant odd-radix OSD systems, $\rho = 2$. From Theorem 2, the comparison constants $C_0$ and $C_1$ are

$$C_0 = -(\alpha - 1) = -\frac{r-1}{2}$$

$$C_1 = r - (\alpha - 1) = \frac{r+1}{2}.$$

Minimally redundant even-radix OSD systems have the following comparison constants (here $\rho = 3$):

$$-(\alpha - 1) = -\frac{r}{2} \leq C_0 \leq \frac{r}{2} + 1$$

$$\frac{r}{2} \leq C_1 \leq \frac{r}{2} + 1 = \alpha.$$

This is slightly different from Avizienis's original algorithm which uses $C_0 = -(\alpha - 1)$ and $C_1 = \alpha$ in all cases, and results in simpler hardware realizations for some values of $r$; including the very important special cases of $r = 2^a$.

Let us take $r = 8$ as an example. The digit set of a minimally redundant radix-8 OSD system is $\{\bar{5}, \cdots, \bar{1}, 0, 1, \cdots, 5\}$. According to Avizienis's OSD addition algorithm, $t_{i+1} = \bar{1}$ for $p_i < -4$ and $t_{i+1} = 1$ for $p_i > 4$. The method presented here allows replacing the second condition by $p_i \geq 4$ which is easier to test in hardware (only the sign bit and a single value bit need to be examined).

Maximally redundant OSD number systems have $\alpha = r - 1$. Let $r > 4$, since for $r = 3$ or $r = 4$, maximally redundant and minimally redundant OSD systems are identical. For such systems, $\rho = r - 1 \geq 3$ and we find the following ranges for the comparison constants:

$$-(r-2) \leq C_0 \leq -1$$

$$2 \leq C_1 \leq r - 1.$$

Selection of values for $C_0$ and $C_1$ is governed by the implementation details. For example, consider the case of $r = 8$. If $p_i$ is formed in sign-and-magnitude or one's complement form, then $C_0 = -3$ and $C_1 = 4$ are the best choices, since the comparisons $p_i \geq -3$ and $p_i \geq 4$ involve the testing of the sign and a single value bit. If negative values are represented in two's complement form, then $C_0 = -4$ and $C_1 = 4$ are the best choices. Note that both cases are different from Avizienis's original OSD proposal which specifies $C_0 = -6$ and $C_1 = 7$ for $r = 8$.

Such changes to the original OSD proposal can also be applied to intermediate OSD systems with $1/2\, r + 1 < \alpha < r - 1$ to obtain similar improvements.

## VI. Stored-Carry Number Systems

Stored-carry number representation systems can be defined for any radix $r$ as having the digit set $\{0, 1, 2, \cdots, r\}$, although only the binary stored-carry (BSC) system has found wide applications. The main application of BSC numbers is in multioperand addition and hence multiplication. A BSC number can be added to a conventional binary number, producing a BSC result, by a set of full adders without carry propagation. The usual encoding for BSC digits in this context is to represent 0 by $\langle 0, 0 \rangle$, 1 by $\langle 0, 1 \rangle$ or $\langle 1, 0 \rangle$, and 2 by $\langle 1, 1 \rangle$. We call this the unary encoding of the digit set $\{0, 1, 2\}$.

That two unary-encoded BSC numbers can be added by a limited-carry circuit consisting of two levels of full adders is well known. This property follows for SC numbers in all radices from our Theorem 5, although the circuit implied by our limited-carry addition procedure is different. In adding two BSC numbers, we have $\lambda^{\min} = 0$ and $\mu^{\min} = 2$ from (10) and (11). To design the needed circuit, we start by selecting appropriate values for $\lambda'$ and $\mu'$ satisfying (28) and (31) which for the BSC number system becomes

$$\lambda' + \mu' \geq 0$$

$$\lambda' \leq -1$$

$$\mu' \leq 1.$$

Clearly, the only possible choices are $\lambda' = -1$ and $\mu' = 1$. The comparison constant $E$ of Theorem 3 must satisfy

$$1 < E \leq 3.$$

The most convenient value for $E$ depends on the encoding used to represent $p_i$. For example, with a 3-bit binary encoding for

$p_i$, the choice $E = 2$ is more convenient since the condition $p_i \geq E$ of Algorithm 4 can be checked by examining the logical OR of the two most significant bits of $p_i$. It is possible to generate the range estimate directly as a function of $x_i$ and $y_i$ (four logic variables) rather than waiting for the computation of $p_i$. This speeds up the addition process at the expense of a somewhat more complex design. If the encoding $\langle 1, 0\rangle$ is disallowed for representing the BSC digit 1, then the choice $E = 3$ turns out to be more convenient and simplifies the overall design considerably. If the unary encoding is used, the absence of $\langle 1, 0\rangle$ can be ensured by adding an initial pair of gates that convert each digit encoding $\langle a_1, a_2\rangle$ to $\langle a_1 a_2, a_1 + a_2\rangle$.

## VII. Stored-Borrow Number Systems

Stored-borrow number representation systems can be defined for any radix $r$ as having the digit set $\{\bar{1}, 0, 1, \cdots, r-1\}$. In the special case of $r = 2$, we obtain the binary stored-borrow (BSB) or binary signed-digit (BSD) number system with the digit set $\{\bar{1}, 0, 1\}$. In addition to having been used for representing temporary values in high-speed multiplication and division [6], [14], [22], [24], BSD numbers have been proposed for application over the entire range of data storage and processing functions in special-purpose arithmetic engines [20]. A BSD number can be added to a conventional binary number, producing a BSD result, by a set of adder-like cells without carry or borrow propagation.

That two BSD numbers can be added by a limited-carry circuit is well known [8], [23]. This property follows for SB numbers in all radices from our Theorem 5, although the circuit implied by our limited-carry addition procedure is different from the previously proposed implementations. In adding two BSB or BSD numbers, we have $\lambda^{\min} = \bar{1}$ and $\mu^{\min} = 1$ from (10) and (11). To design the needed circuit, we start by selecting appropriate values for $\lambda'$ and $\mu'$ satisfying (28) and (31) which for the BSB number system become

$$\lambda' + \mu' \geq 0$$

$$\lambda' \leq 0$$

$$\mu' \leq 0.$$

Clearly, the only possible choices are $\lambda' = \mu' = 0$. The comparison constant $E$ of Theorem 3 must satisfy

$$-1 < E \leq 1.$$

The most convenient value for $E$ depends on the encoding used to represent $p_i$. In most cases, however, the choice $E = 0$ is convenient since the condition $p_i \geq E$ of Algorithm 4 can be checked by determining the sign of $p_i$. It is also possible to generate the range estimate directly as a function of $x_i$ and $y_i$ (four logic variables) rather than waiting for the computation of $p_i$, as was the case for BSC numbers.

In two-valued logic, each binary signed digit can be represented by two bits, using several possible encodings. Two natural encodings are

1) the $\langle s, v\rangle$ encoding, consisting of "sign" and "value" bits for each digit, whereby $\bar{1}$, 0, and 1 are represented by $\langle 1, 1\rangle$, $\langle 0, 0\rangle$, and $\langle 0, 1\rangle$, respectively,

2) the $\langle n, p\rangle$ encoding, consisting of "negative" and "positive" flags for each digit, whereby $\bar{1}$, 0, and 1 are represented by $\langle 1, 0\rangle$, $\langle 0, 0\rangle$, and $\langle 0, 1\rangle$, respectively.

If a digit $d$ is represented as $\langle d^s, d^v\rangle$ with the first and as $\langle d^n, d^p\rangle$ with the second encoding, then the following equalities hold:

$$d = (1 - 2d^s)d^v = d^p - d^n.$$

It has been shown that the second encoding results in much simpler implementations for most arithmetic circuits of interest [20]. Both encodings allow the implementation of normalized significant digit arithmetic [15] if the extra combination ($\langle 1, 0\rangle$ in the $\langle s, v\rangle$ encoding and $\langle 1, 1\rangle$ in the $\langle n, p\rangle$ encoding) is used to denote nonsignificant zeros. On the other hand, the extra combination $\langle 1, 0\rangle$ of the $\langle s, v\rangle$ encoding and the unused combination $\langle 1, 1\rangle$ of the $\langle n, p\rangle$ encoding can be used as DON'T CARE conditions to obtain simpler designs. It is also possible to use a 1-out-of-3 encoding where a binary signed digit is represented by the triple $\langle n, o, p\rangle$, with the middle flag denoting the value zero. Such an encoding can provide complete unidirectional error detection capability with a relatively low overhead in terms of added hardware complexity.

An important property of BSD numbers is that there exists a propagation-free recoding algorithm which transforms any BSD number $x = x_{k-1}x_{k-2} \cdots x_0$ into an equivalent BSD number $z = z_k z_{k-1} \cdots z_0$, such that $z_j \cdot z_{j-1} \neq 1$ $(1 \leq j \leq k)$. This recoding enables the use of a special carry-free addition process instead of the limited-carry process and also leads to higher-speed serial or parallel multiplication. The details can be found in [20] and thus will not be discussed here.

As BSD numbers require the same amount of storage as the stored-carry representation (2 bits per digit position), it is natural to ask whether BSD numbers offer any advantage over the BSC system. The answer is positive for the following reasons:

1) ease of multiplication, division, and other arithmetic operations,
2) ease of zero detection,
3) suitability for use with arithmetic error codes.

These points have been dealt with elsewhere [20].

## VIII. Stored-Carry-or-Borrow Number Systems

The stored-carry and stored-borrow properties can be combined to obtain the SCB number representation systems which in radix $r$ use the digit set $\{\bar{1}, 0, 1, \cdots, r\}$. Unfortunately, despite the higher redundancy index ($\rho = 2$) compared to the SC and SB systems ($\rho = 1$), carry-free addition is still not possible because we have $\alpha = 1$ (refer to Theorem 1). Furthermore, other arithmetic algorithms (such as multiplication) can only become more difficult due to the extra digit value. It is therefore quite surprising for SCB number systems to find any application at all.

One possible application area for SCB number systems is in the design of systolic up/down counters. The binary SCB (BSCB) number system was first used in the design of a systolic up/down counter in 1982 [9], apparently without any attention to its arithmetic properties. A later systolic binary counter design by the author [17] also used the BSCB number

system in a different disguise. It was this application of the BSCB number system that prompted the author to study the properties of GSD number systems.

In the systolic binary counter application, the magnitude of the count can be stored as a BSCB number, with the sign maintained separately. Assuming that the count value 0 is detectable without a need for signal propagation and that the value 0 is always given the positive sign, then the various counter operations can be performed as follows.

1) To count up from a nonnegative value, increment the magnitude.

2) To count up from a negative value, decrement and change the sign if the result is 0.

3) To count down from a nonnegative value, first change the sign if the starting value is 0 and then decrement.

4) To count down from a negative value, increment the magnitude.

Increment and decrement operations affect only the least significant counter position directly. At each clock pulse, the value of the $i$th counter digit is recomputed based on its current value and the value of the $(i-1)$th digit according to a simple set of rewriting rules which essentially transfer a stored carry or borrow from one position to the next higher position, enabling the absorption of the incoming carry or borrow (increment or decrement) at the least significant position.

This method is also applicable to higher radix systolic counters which have the advantage of lower overhead. In particular, for $r = 10$, no storage overhead is involved since a decimal counter requires at least 4 bits per digit position anyway. The logic of each counter stage is of course more complex for a systolic decimal counter than for an ordinary counter and this is the price one pays for the added speed.

## IX. Conclusions

We have presented a unified framework for the study of redundant number representation systems, providing a general theory for carry-free and limited-carry addition properties of such systems. In fact, carry-free addition can be considered a special case of limited-carry addition where the estimate $e_i$ is not needed because we have $\lambda' = \lambda$ and $\mu' = \mu$. So, we could have chosen to present the limited-carry results first and then derive the carry-free property as a special case by seeking conditions under which the equalities $\lambda' = \lambda$ and $\mu' = \mu$ can be satisfied. However, it is felt that the present order of presentation is more natural and easier to understand.

Does the generalization considered in this paper suggest new classes of redundant number representations which have not been dealt with in the past? The answer to this question is positive. An immediate example is provided by the SCB subclass which has been previously examined in a limited context and only for $r = 2$. Consider as another example an interesting subclass of GSD number systems which is obtained if we choose $\alpha = 0$ and $\beta = r + 1$. This subclass, which may be called the stored-double-carry (SDC) number system, implies the same amount of storage overhead and redundancy as the SCB subclass but offers the advantage of carry-free addition. The obvious disadvantage with respect to SCB systems is the use of digits with larger magnitudes and thus the potential for more difficult multiplication and division. The suitability of such new number systems must be evaluated in the context of particular application areas. However, one can immediately conclude that SDC number systems are attractive for applications where add/subtract operations are dominant.

Other potentially useful subclasses are decimal number systems with higher redundancy indexes than the decimal SDC system. For example, the hex-digit decimal (HDD) number system with the digit set $\{0, 1, \cdots, 15\}$ implies no storage overhead compared to conventional decimal representation but offers the advantage of carry-free addition using a two-stage circuit (an ordinary hex adder and a "correction" circuit which is only slightly more complex than the conventional decimal "adjuster"). Compared to low-redundancy OSD number systems (digit set $\{\bar{6}, \cdots, 0, \cdots, 6\}$ or $\{\bar{7}, \cdots, 0, \cdots, 7\}$), the HDD representation has the advantages of no initial conversion from decimal and simpler adder logic and the disadvantage of needing an extra sign bit. The properties of this and other new redundant decimal number systems are currently under investigation.

In general, the use of redundant number representation systems is particularly effective when long sequences of computations and/or long operands are involved. Such conditions prevail in many special-purpose systems such as those used for high-precision scientific computations and signal processing [11], [26], [10], [13]. Further work is thus needed to identify strategies for the evaluation of various redundant representations with respect to suitability for particular classes of applications.

Finally, from the point of view of practical implementation, our discussion in this paper is incomplete due to inadequate treatment of subtraction as well as zero detection and sign test for GSD number representations. Because GSD number systems may have asymmetric digit sets, we need to consider subtraction (or at least sign change for representations with $\alpha > 0$ and $\beta > 0$) explicitly. Most results presented here carry over directly to subtraction, using the *position difference* $q_i = x_i - y_i$, the *interim difference* $u_i = q_i + rt_{i+1}$, and the *final difference* digit $d_i = u_i - t_i$ in the algorithms. Details of the procedures and related results along with methods for zero detection and sign test have been worked out in a companion paper [21].

## References


[1] S. F. Anderson *et al.*, "The IBM System/360 Model 91: Floating-point execution unit," *IBM J. Res. Develop.*, pp. 34-53, Jan. 1967.

[2] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," *IRE Trans. Comput.*, vol. EC-10, pp. 389-400, 1961.

[3] ——, "On a flexible implementation of digital computer arithmetic," in *Inform. Processing '62*. Amsterdam: North-Holland, 1963, pp. 664-670.

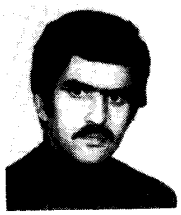
[4] ——, "Binary-compatible signed-digit arithmetic," in *AFIPS Conf. Proc. (1964 Fall Joint Comput. Conf.)*, pp. 663-672.

[5] ——, "Arithmetic error codes: Cost and effectiveness studies for applications in digital system design," *IEEE Trans. Comput.*, vol. C-20, no. 11, pp. 1322-1331, Nov. 1971.

[6] A. D. Booth, "A signed binary multiplication technique," *Quarterly J. Mech. Appl. Math.*, vol. 4, part 2, pp. 236-240, 1951.

[7] T. C. Chen, "Maximal redundancy signed-digit systems," in *Proc. Symp. Comput. Arithmetic*, Urbana, IL, June 1985, IEEE Computer Society Press, pp. 296-300.

[8] C. Y. Chow and J. E. Robertson, "Logical design of a redundant binary adder," in *Proc. Symp. Comput. Arithmetic,* Santa Monica, CA, Oct. 1978, pp. 109–115.
[9] L. J. Guibas and F. M. Liang, "Systolic stacks, queues, and counters," in *Proc. Conf. Advanced Res. VLSI,* MIT, 1982, pp. 155–164.
[10] L. B. Jackson *Digital Filters and Signal Processing.* Boston, MA: Kluwer, 1986.
[11] R. W. Hockney and C. R. Jesshope, *Parallel Computers.* Bristol, England: Adam Hilger, 1981.
[12] D. E. Knuth, *The Art of Computer Programming: Vol. 2 Seminumerical Algorithms*, 2nd ed. Reading MA: Addison-Wesley, 1981, pp. 179–197.
[13] S.-Y. Kung, R. E. Owen, and J. G. Nash, Eds., *VLSI Signal Processing II.* New York: IEEE Press, 1986.
[14] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proc. IRE,* vol. 49, pp. 67–91, Jan. 1961.
[15] N. Metropolis and R. L. Ashenhurst, "Significant digit computer arithmetic," *IRE Trans. Electron. Comput.,* vol. EC-7, pp. 265–267, 1958.
[16] G. Metze and J. E. Robertson, "Elimination of carry propagation in digital computers," in *Inform. Processing '59 (Proc. UNESCO Conf.,* June 1959), 1960, pp. 389–396.
[17] B. Parhami and A. Avizienis, "Detection of storage errors in mass memories using low-cost arithmetic error codes," *IEEE Trans. Comput.,* vol. C-27, no. 4, pp. 302–308, Apr. 1978.
[18] B. Parhami, "Systolic up/down counters with zero and sign detection," in *Proc. Symp. Comput. Arithmetic,* Como, Italy, May 1987, pp. 174–178.
[19] ——, "A general theory of carry-free and limited-carry computer arithmetic," in *Proc. Canadian Conf. VLSI,* Winnipeg, Canada, Oct. 1987, pp. 167–172.
[20] ——, "Carry-free addition of recoded binary signed-digit numbers," *IEEE Trans. Comput.,* pp. 1470–1476, Nov. 1988.
[21] ——, "On the practical implementation of arithmetic functions with generalized signed-digit number representation," submitted for publication.
[22] J. E. Robertson, "A new class of digital division methods," *IEEE Trans. Comput.,* vol. C-7, pp. 218–222, Sept. 1958.
[23] N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Comput.,* vol. C-34, no. 9, pp. 789–796, Sept. 1985.
[24] K. D. Tocher, "Technique for multiplication and division for automatic binary computers," *Quarterly J. Mech. Appl. Math.,* vol. 11, part 3, pp. 364–384, 1958.
[25] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.,* vol. EC-14, no. 1, pp. 14–17, Feb. 1964.
[26] V. Zakharov, "Parallelism and array processing," *IEEE Trans. Comput.,* vol. C-33, no. 1, pp. 45–78, Jan. 1984.



**Behrooz Parhami** (S'70–M'73–SM'78) was born on February 1, 1947 in Tehran, Iran. He received B.S. and M.S. degrees in electrical engineering from Tehran University and Oregon State University, in 1968 and 1970, respectively, and the Ph.D. degree in computer science from University of California at Los Angeles in 1973.

He was a Research Assistant/Engineer (1970–1973) and Acting Assistant Professor (1973–1974) at UCLA before joining Sharif (then Arya-Mehr) University of Technology (SHUT) in Tehran, Iran, where he was Professor of Computer Science until 1988. During his tenure at SHUT, he was active in the areas of educational planning and curriculum development and also taught a wide variety of courses. This led to his involvement in the development of a standard nationwide computer science and engineering program which has been in effect in all major Iranian universities since 1983. Since 1972, he has acted as a consultant to various organizations, been involved in standardization activities, participated in many conference organizing and program committees, and served on the editorial and advisory boards of several technical publications (including a five-year term as Editor of *Computer Report,* a Farsi-language computing periodical). In 1986–1988, he was on sabbatical leave as Visiting Professor at the University of Waterloo and Carleton University in Canada, before joining University of California at Santa Barbara as Professor of Computer Engineering in 1988. Dr. Parhami has published more than 60 research papers in the areas of computer architecture, computer arithmetic, dependable computing, Farsi-language information processing, and informatics education. He has also authored two Farsi-language texts entitled *Computer Appreciation* and *Computer Organization, Vol. 1: Basics of Hardware* and has coauthored a 6000-term *English–Farsi Glossary of Computers and Informatics.* His current research interests are in the general areas of computer hardware and organization, including high-performance systems and dependability issues.

Dr. Parhami is a Distinguished Member of the Informatics Society of Iran (ISI), and a member of the Association for Computing Machinery, and the British Computer Society. He was the founder of ISI and served as its first president from 1978 to 1983. He also served as Chairman of IEEE Iran Section from 1977 to 1986, following a one-year term as Vice-Chairman and received the IEEE Centennial Medal in 1984.