Implementation Alternatives for Generalized Signed-Digit Addition

Behrooz Parhami

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

Abstract

Even though the theory of carry-free and limited-carry generalized signed-digit addition has been known for some time, exploration of the vast design space encompassing the number representation radix, choice of a redundant digit set, consideration of all possible encodings for operand and transfer digits, and selection of values for the various "free" parameters in the general algorithm has just begun. In this paper, we review several implementation alternatives and place them into a unified framework that provides insight, facilitates comparisons and tradeoffs, helps future designers in effectively exploring the design space, and leads to novel representations or algorithms in some cases.

Keywords: Carry-free addition, High-radix arithmetic, Redundant number systems, Signed-digit representation.

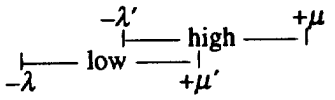
1. Introduction

A generalized signed-digit (GSD) number system is a fixed-radix positional representation utilizing the digit set $[-\alpha, \beta] = \{-\alpha, -\alpha+1, \ldots, \beta-1, \beta\}$ in radix r with $\alpha \geq 0$, $\beta \geq 0$, and $\rho = \alpha + \beta + 1 - r$, where $\rho \geq 1$ is the redundancy index of the number system [PARH87a], [PARH88], [PARH88a], [PARH90], [PARH93].

It has been shown that any GSD number system with $r>2$ and $\rho \geq 3$ (or with $\rho=2$, provided that $\alpha \neq 1$ and $\beta \neq 1$) supports carry-free arithmetic and that even in the exceptional cases (i.e., for $r = 2$, $\rho = 1$, or $\rho = 2$ with $\alpha=1$ or $\beta = 1$) a limited-carry addition/subtraction algorithm is applicable which yields the ith sum digit s_i as a function of the digits x_i, y_i, x_{i-1}, y_{i-1}, x_{i-2}, and y_{i-2} of the two operands x and y. The carry-free algorithm utilizes transfer digits in the range $[-\lambda, \mu]$ which go to the next higher position and are incorporated there without propagating further.

Carry-free addition is possible, for example, in the case of radix-4 redundant representations with digit sets [–3, 3], [–3, 2], [–2, 3], and [0, 5], among others.

The limited-carry addition algorithm uses an additional two-valued "transfer estimate" parameter $e_i \in$ {low, high} whose arrival precedes the actual transfers.



The function of the information carried by e_i is to exclude one extreme in the range of incoming transfers and thus to enable the transfer generation logic to pick a value for the outgoing transfer digit in such a way that it guarantees the capacity to absorb the incoming transfer without exceeding the allowed range of digit values. For example, if e_i = low, then w_i can safely exceed $\beta - \mu$, possibly going as high as $\beta - \mu'$, without a problem. This added slack, it turns out, is adequate for covering all the special cases that do not support carry-free addition.

Examples of GSD systems in the limited-carry category are the stored-carry, stored-borrow, and stored-carry-or-borrow representation methods corresponding to the digit sets $[0, r]$, $[-1, r-1]$, and $[-1, r]$, respectively, in radix r, including their important radix-2 special cases, and radix-4 representation with digit set $[-1, 4]$. Stored-carry, stored-borrow, and stored-carry-or-borrow number systems have found numerous practical applications in the design of high-speed arithmetic circuits with both conventional and redundant representations (see, for example, [PARH87], [PARH88b], [PARH89a], [PARH94]).

Application of GSD arithmetic to the design of fast circuits and algorithms for signal processing and other applications is well documented in the literature. Proposed implementations range from pipelined digit-serial (or on-line) to fully parallel units, with solutions having various degrees of digit-parallel operation falling between the two extremes [ERCE89], [HUNG93], [PARH92].

Recently, many implementations of GSD arithmetic have been attempted from different perspectives and with varying objectives, environments, and technological constraints. Examples include optical arithmetic processors [AWWA92], [CHER88], processors exploiting multi-valued logic technologies [ETIE93], [HOHL90], [KAME90], [KAWA91], [KAWA92], [KAWA94], [MICH92], [MURA93], and hybrid (mix of redundant and non-redundant digit positions) realizations for optimizing VLSI parameters [PHAT94]. Typically, implementers start from scratch, developing the needed theory and selecting the various parameters introduced in the

1058-6393/95 $4.00

theoretical development in view of performance requirements and technological constraints.

In this paper, it is shown that all these seemingly different systems and their associated algorithms are actually "implementations" of the GSD theory developed by this author. These variations correspond to various radices r, digit sets (parameters α and β), transfer values (parameters λ and μ) and encodings of these sets using binary or multi-valued signals. This characterization:

1. Unifies the various implementations under a single theoretical framework that provides insight and also facilitates comparisons and tradeoff studies.
2. Clearly shows that there are other choices for the parameters involved; these alternatives may not be evident with ad-hoc development.
3. Facilitates the development of appropriate redundant number representations to take advantage of new and emerging implementation technologies.
4. Allows the derivation of novel representation systems and associated algorithms by merely experimenting with the various implementation parameters.

Examples of novel redundant representations are provided along with their associated arithmetic algorithms and implementation alternatives. These representations are suitable for the implementation of arithmetic algorithms with both binary and multi-valued logic (MVL).

2. Carry-Free Addition

Ideally, the term "carry-free addition" should be applied to the case where the sum digit s_i is only a function of the operand digits x_i and y_i (in this paper, we use 0-origin, right-to-left indexing). This is clearly impossible for positional number representations. It appears to hold for residue number system (RNS) representations [SODE86], but the size of "residue digits" to provide a given range M is at least logarithmic in M, thus leading to some form of carry propagation within residue digits, even though the residues are processed independently.

Given the impossibility of pure digit-by-digit carry-free addition, the term carry-free has traditionally been applied to the next best thing: i.e., the addition/subtraction scheme defined in Algorithm 1 where position $i-1$ produces a transfer digit t_i that goes to position i and is absorbed there without causing further propagation. Figure 1 provides a graphical representation of the two-stage carry-free addition process defined by Algorithm 1. Any type of transfer propagation beyond this absolute minimum (e.g., 2 stages of propagation, as depicted in Figures 3 and 4) will be referred to as "limited-carry" in this paper.

Algorithm 1: Two-Stage Carry-Free GSD Addition

1. Compute the position sum $p_i = x_i + y_i$. Decompose p_i into the transfer digit t_{i+1} and interim sum digit w_i such that $p_i = rt_{i+1} + w_i$.
2. Compute the sum digit $s_i = w_i + t_i$. ■

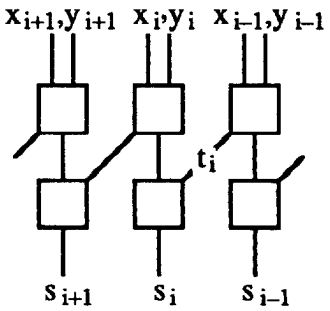


Figure 1. Two-stage carry-free addition process.

Note that the scheme shown in Figure 1 is slightly more general that the one given by Algorithm 1 in that the production of t_i in Figure 1 need not be based on the explicit computation of $p_i = x_i + y_i$, leading to more flexibility and higher speed in some cases.

To further elaborate on this point, we note that the function of t_i is to convey some information about position $i-1$ to position i. The needed information can in fact be conveyed by the operand digits themselves or by partial information about the operand digits (e.g., a subset of the bits used to represent the digits in binary or only their signs). Hence, the single-stage addition process, defined in Algorithm 2 and depicted in Figure 2, is also feasible in some cases.

Algorithm 2: Single-Stage Carry-Free GSD Addition

Compute the sum digit $s_i = f(x_i, y_i, g(x_{i-1}), h(y_{i-1}))$. ■

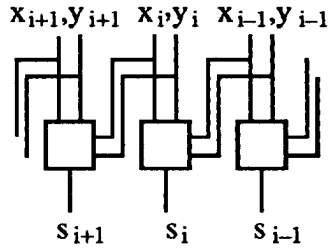


Figure 2. Single-stage carry-free addition process.

For Algorithm 2 to be truly a single-stage process, g and h must be simple functions that require no computation. Clearly, one can use the identity function for both g and h,

but this may lead to a large number of inputs for the logic function f with binary implementation or may make f hard to realize with MVL. It would be better if g and h corresponded to selecting a subset of logic signals representing x_{i-1} and y_{i-1}. Clearly, intermediate schemes between those of Figure 1 and Figure 2 are possible in which g and h are very simple functions (perhaps needing a single gate level with binary logic or have an equivalently simple realization with MVL).

The choice between Algorithms 1 and 2 (Figures 1 and 2) is dependent on both the number system being used and the implementation technology. As an example, values can be added quite simply with multiple-valued current-summing logic. This makes Algorithm 1 quite attractive since obtaining the position sum p_i in the first stage and the entire second stage become trivial.

A subclass of the recently introduced hybrid signed-digit (HSD) number systems can be viewed as an implementation scheme for GSD addition with certain radices and digit sets. In this subclass of HSD numbers, a kd-digit number uses ordinary binary digits except in positions $d-1, 2d-1, \ldots, kd-1$ where the digit set $[-1, 1]$ is used. Such numbers can be viewed as radix-2^d GSD numbers with the digit set $[-2^{d-1}, 2^d-1]$. It is easy to show that the transfer $t_{i+1} \in [-1, 1]$ is only a function of the three most significant bits in the binary representation of each radix-2^d operand (2 bits for the signed digit position and 1 bit for the next lower position).

3. Limited-Carry Addition

The limited carry process based on preliminary estimates of transfer digit values, described in Section 1, is formalized in Algorithm 3. One way to compute e_{i+1} based on p_i is to compare p_i to a known constant [PARH90], but simpler schemes may also be possible. Figure 3 provides a graphical representation of the 3-stage limited-carry addition process defined by Algorithm 3.

Algorithm 3: Three-Stage Limited-Carry GSD Addition Based on Transfer Digit Estimates

1. Compute the position sum $p_i = x_i + y_i$. Based on p_i, generate the estimate $e_{i+1} \in$ {low, high} for t_{i+1}.

2. Based on the incoming estimate e_i, decompose p_i into the transfer digit t_{i+1} and interim sum digit w_i such that $p_i = rt_{i+1} + w_i$.

3. Compute the sum digit $s_i = w_i + t_i$. ■

Again, the scheme shown in Figure 3 is somewhat more general than the one given by Algorithm 3 in that the production of e_i or even t_i in Figure 3 need not be based on the explicit computation of $p_i = x_i + y_i$.

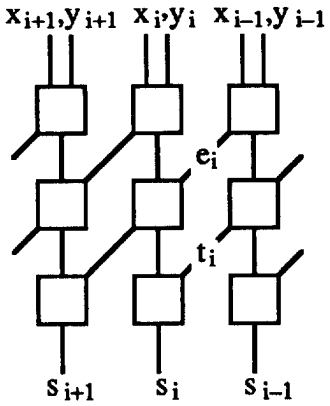


Figure 3. Three-stage limited-carry addition process with transfer digit estimates e_i.

A possible alternative to Algorithm 3 for limited-carry GSD addition is using repeated or double transfers. This is defined in Algorithm 4 and Figure 4.

Algorithm 4: Three-Stage Limited-Carry GSD Addition Based on Repeated or Double Transfers

1. Compute the position sum $p_i = x_i + y_i$. Decompose p_i into the first transfer digit t'_{i+1} and the initial interim sum digit w'_i such that $p_i = rt'_{i+1} + w'_i$.

2. Compute the initial sum digit $s'_i = w'_i + t'_i$. Decompose s'_i into the second transfer digit t_{i+1} and the final interim sum digit w_i such that $s'_i = rt_{i+1} + w_i$.

3. Compute the final sum digit $s_i = w_i + t_i$. ■

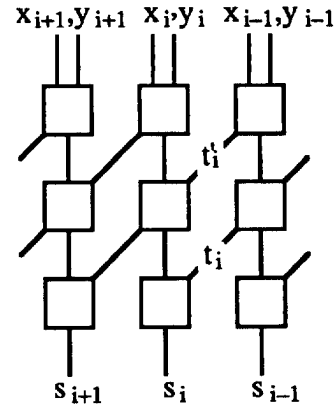


Figure 4. Three-stage limited-carry addition process with repeated or double transfers.

The two transfers do not have to be produced by identical mechanisms or even have the same ranges. But if this were the case, then Algorithm 4 would offer no advantage over Algorithm 3, since the generation of e_i is simpler than that of t'_{i+1} plus w'_i. The primary advantage of

Figure 4 over Figure 3 is its more regular design for VLSI or sequential realization of the first and the second stage, with shared hardware, for a lower-cost implementation.

As in Section 2, we note that the e_i (t'_i) and t_i collectively convey some information about positions $i-2$ and $i-1$ to position i. The needed information can in fact be conveyed by the operand digits themselves or, again, by partial information about the operand digits. Hence, the single-stage addition process depicted in Figure 2 can be modified, by adding connections from x_{i-2} and y_{i-2} to the logic in Stage i, to handle this case. However, this is likely to lead to highly complex stage logic.

As a compromise, one can use the two-stage parallel-transfers approach of Algorithm 5 in which Position i sends two transfers, $t_{i+1}{}^{(1)}$and $t_{i+2}{}^{(2)}$, to Positions $i+1$ and $i+2$, respectively. Note again that the graphical depiction of Figure 5 can be viewed as being somewhat more general than Algorithm 5.

Algorithm 5: Two-Stage Carry-Free GSD Addition Based on Parallel Transfers

1. Compute the position sum $p_i = x_i + y_i$. Decompose p_i into two transfer digits $t_{i+2}{}^{(2)}$, $t_{i+1}{}^{(1)}$ and the interim sum digit w_i such that $p_i = r^2 t_{i+2}{}^{(2)} + r t_{i+1}{}^{(1)} + w_i$.
2. Compute the final sum digit $s_i = w_i + t_i{}^{(1)} + t_i{}^{(2)}$. ■

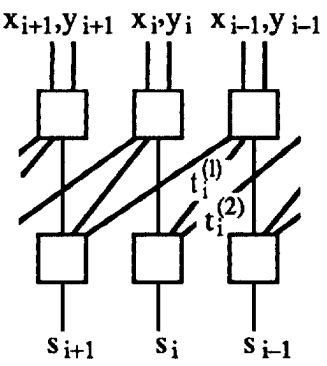


Figure 5. Two-stage carry-free addition process with parallel carries $t_i{}^{(1)}$ and $t_i{}^{(2)}$.

As an example, two-stage carry-free addition is possible for radix-2 numbers using the digit set [0, 3] (stored-double-carry numbers) or [0, 4] (stored-triple-carry numbers). These particular unsigned-digit redundant representations were first introduced in [PARH89] in connection with the design of high-speed multipliers that have area-efficient layouts in view of their binary-tree structures and were used more recently for MVL realization of such multipliers in [KAWA91], [KAWA92], [KAWA94]. In both of these cases, the transfers are in [0, 1].

The use of parallel carries can be generalized to include more than two receiving positions, non-binary transfer digits, and non-uniform transfer digit sets. Denoting the transfer going from Position $i-j$ to Position i by $t_i{}^{(j)}$, with $1 \le j \le h$ and $t_i{}^{(j)} \in [-\lambda_j, \mu_j]$, one can derive from the basic addition equation

$$x_i + y_i + \sum_{j=1}^{h} t_i{}^{(j)} = \sum_{j=1}^{h} r^h t_{i+h}{}^{(j)} + s_i$$

and the various ranges assumed, the relevant constraints to be met. Some results with uniform transfers, i.e. $\lambda_j = \lambda$ and $\mu_j = \mu$ ($1 \le j \le h$), can be found in [PARH94a].

However, there is no compelling reason for uniformity in transfer digit sets (see [KORN94] for an illuminating discussion of digit sets). One possibility is to have transfers that alternate in sign. For the sake of simplicity, assume $\lambda_j = \gamma \times (j \bmod 2)$ and $\mu_j = \gamma \times (1 - j \bmod 2)$, leading to $\lambda_j + \mu_j = \gamma$. Then, each position i receives $\lfloor h/2 \rfloor$ positive and $\lceil h/2 \rceil$ negative transfers, each up to γ in magnitude. Hence the interim sum must be in the range $-\alpha + \gamma\lceil h/2 \rceil \le w_i \le \beta - \gamma\lfloor h/2 \rfloor$ if all transfers are to be absorbed without exceeding the allowed digit range $[-\alpha,\beta]$. Since w_i should assume at least r different values, a necessary condition is:

$$\beta + \alpha - h\gamma + 1 \ge r \quad \text{or} \quad \rho \ge h\gamma$$

Radices $r > 2$ impose additional restrictions, but for $r = 2$ and $\gamma = 1$, digit sets such [–1, 2] with h=2 (corresponding to the stored-carry-or-borrow representation [PARH87]) and [–3, 1] with $h = 3$, as well as their mirror images [–2, 1], [–1, 3] obtained by letting $\mu_j = \gamma \times (j \bmod 2)$, are quite practical. Such digit sets involve smaller maximum magnitudes than the unsigned versions, thus leading to potential simplifications in various arithmetic operations such as multiplication, division, and square-rooting.

4. Conclusion

We have reviewed several implementation alternatives for carry-free and limited-carry GSD addition and placed them into a unified framework that provides insight, facilitates comparisons and tradeoffs, helps future designers in effectively exploring the design space, and leads to novel representations or algorithms in some cases.

With renewed interest in GSD representations, spurred primarily by advances in optical and MVL logic technologies, researchers in these fields seem to be reinventing much of the theory already developed in the field of computer arithmetic. It is hoped that the unified view and systematic approach to choosing implementation alternatives and associated parameters for GSD addition, advocated in this paper, leads to better understanding and communication between researchers in these fields.

References

[AWWA92] Awwal, A.A., *Applied Optics*, pp. 3205-3208, 10 June 1992.

[CHER88] Cherri, A.K. and M.A. Karim, "Modified Signed Digit Arithmetic Using an Efficient Symbolic Substitution Logic", *Applied Optics*, Vol. 27, No. 18, pp. 3824-3827, 1988.

[ERCE89] Ercegovac, M.D. and T. Lang, "On-Line Arithmetic for DSP Applications", *Proc. Midwest Symp. on Circuits and Systems*, Aug. 1989, pp. 365-368.

[ETIE93] Etiemble, D. and K. Navi, "A Basis for the Comparison of Binary and m-Valued Current Mode Circuits: The Multioperand Addition with Redundant Number Systems", in [SMVL93], pp. 216-221.

[HOHL90] Ho, H.L. and K.C. Smith, "Integrator-Chain Multiplier", in [SMVL90], pp. 363-369.

[HUNG93] Hung, C.Y. and B. Parhami, "Generalized Signed-Digit Multiplication and Its Systolic Realizations", *Proc. Midwest Symp. Circuits & Systems*, Detroit, Aug. 1993, pp. 1505-1508.

[KAME90] Kameyama, M., M. Nomura, and T. Higuchi, "Modular Design of Multiple-Valued Arithmetic VLSI System Using Signed-Digit Number System", in [SMVL90], pp. 355-362.

[KAWA91] Kawahito, S., K. Mizuno, and T. Nakamura, "Multiple-Valued Current-Mode Arithmetic Circuits Based on Redundant Positive-Digit Number Representations", in [SMVL91], pp. 330-339.

[KAWA92] Kawahito, S., Y. Mitsui, M. Ishida, and T. Nakamura, "Parallel Hardware Algorithms with Redundant Number Representations for Multiple-Valued Arithmetic VLSI", in [SMVL92], pp. 337-345.

[KAWA94] Kawahito, S., M. Ishida, T. Nakamura, M. Kameyama, and T. Higuchi, "High-Speed Area-Efficient Multiplier Design Using Multiple-Valued Current-Mode Circuits", *IEEE Transactions on Computers*, Vol. 43, No. 1, pp. 34-42, Jan. 1994.

[KORN94] Kornerup, P., "Digit-Set Conversions: Generalizations and Applications", *IEEE Transactions on Computers*, Vol. 43, No. 5, pp. 622-629, May 1994.

[MICH92] Micheel, L.J., "Heterojunction Bipolar Technology for Emitter-Coupled Multiple-Valued Logic in Gigahertz Adders and Multipliers", in [SMVL92], pp. 18-26.

[MURA93] Muranaka, N., S. Imanishi, and D.M. Miller, "Decimal Addition and Subtraction Units Using the p-Valued Decimal Signed-Digit Number Representation", in [SMVL93], pp. 228-233.

[PARH87] Parhami, B., "Systolic Up/Down Counters with Zero and Sign Detection", *Proc. of the Int'l Symp. on Computer Arithmetic*, Como, Italy, May 1987, pp. pp. 174-178.

[PARH87a] Parhami, B., "A General Theory of Carry-Free and Limited-Carry Computer Arithmetic", *Proc. of the Canadian Conf. on VLSI*, Winnipeg, Canada, Oct. 1987, pp. 167-172.

[PARH88] Parhami, B., "Conversions Between Generalized Signed-Digit and Conventional Number Representations", *Proc. of the 2nd Int'l Parallel Processing Symp.*, Fullerton, CA, Apr. 1988, pp. 95-106.

[PARH88a] Parhami, B., "Zero, Sign, and Overflow Detection Schemes for Generalized Signed-Digit Arithmetic", *Proc. of the 22nd Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Oct./Nov. 1988, pp. 636-639.

[PARH88b] Parhami, B., "Carry-Free Addition of Recoded Binary Signed-Digit Numbers", *IEEE Transactions on Computers*, Vol. 37, No. 11, pp. 1470-1476, Nov. 1988.

[PARH89] Parhami, B., "A New Method for Designing Highly Parallel Binary Multipliers", *Proc. of the 3rd Int'l Parallel Processing Symp.*, Fullerton, CA, Mar. 1989, pp. 176-185.

[PARH89a] Parhami, B., "Parallel Counters for Signed Binary Signals", *Proc. of the 23rd Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Oct. 1989, pp. 513-516.

[PARH90] Parhami, B., "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations", *IEEE Transactions on Computers*, Vol. 39, No. 1, pp. 89-98, Jan. 1990.

[PARH92] Parhami, B., "Flexible Massively Parallel Arithmetic on Associative Processors", *Preprints of the Associative Processing and Applications Workshop*, Syracuse, NY, July 1992, pp. 16-1 to 16-8.

[PARH92a] Parhami, B., "Systolic Number Radix Converters", *The Computer Journal*, Vol. 35, No. 4, pp. 405-409, Aug. 1992.

[PARH93] Parhami, B., "On the Implementation of Arithmetic Support Functions for Generalized Signed-Digit Number Systems", *IEEE Transactions on Computers*, Vol. 42, No. 3, pp. 379-384, Mar. 1993.

[PARH94] Parhami, B., "Comments on 'Evaluation of $A + B = K$ Conditions Without Carry Propagation'", *IEEE Transactions on Computers*, Vol. 43, No. 3, p. 381, Mar. 1994.

[PARH94a] Parhami, B., "Comments on 'High-Speed Area-Efficient Multiplier Design Using Multiple-Valued Current Mode Circuits'", *IEEE Transactions on Computers*, to appear.

[PHAT94] Phatak, D.S. and I. Koren, "Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains", *IEEE Transactions on Computers*, Vol. 43, No. 8, pp. 880-891, Aug. 1994.

[SMVLyr] *Proc. of the Int'l Symp. on Multiple-Valued Logic*, Annual, usually in May.

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