Constant-Time Addition with Hybrid-Redundant Numbers: Theory and Implementations

Ghassem Jaberipur
Department of Electrical and Computer Engineering
Shahid Beheshti University, Tehran 19839-63113, Iran, and
School of Computer Science
Institute for Studies in Theoretical Physics and Mathematics (IPM)
E-mail: jaberipur@sbu.ac.ir

Behrooz Parhami
Corresponding author
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560, USA
Phone +1 805 893 3211, Fax +1 805 893 3262
E-mail: parhami@ece.ucsb.edu

Abstract: Hybrid-redundant number representation has provided a flexible framework for digit-parallel addition in a manner that facilitates area-time tradeoffs for VLSI implementations via arbitrary spacing of redundant digit positions within an otherwise nonredundant representation. We revisit the hybrid redundancy scheme, pointing out limitations such as representational asymmetry, lack of representational closure in certain adder implementations, and difficulties in subtraction and carry acceleration. Given the intuitiveness of the hybrid redundancy concept and its potential for describing practically useful redundant number systems, we are motivated to extend it within the framework of weighted bit-set encodings to circumvent the aforementioned problems. The extension is based mainly on allowing negatively weighted bits (negabits), as well as standard posibits, to appear in nonredundant positions. Our extended hybrid redundancy scheme provides for arbitrary spacing of redundant positions in symmetric digit sets, without any degradation in arithmetic efficiency, while at the same time allowing low-latency subtraction by means of the same circuitry that is used for addition. Finally, we describe how inverted encoding of negabits leads to the exclusive use of unmodified standard full/half-adder, counter, and compressor cells, with no extra inverters, and to the direct applicability of conventional carry acceleration techniques in constant-time addition.

Keywords: Arithmetic unit, carry-free addition, computer arithmetic, redundant number representation, signed-digit number system.

1. Introduction

Implementation of arithmetic algorithms has been subject to continual improvement to allow greater speed and to reduce VLSI area and power. The choice of number system and its encoding has a major influence on achieving such design goals. For example, the three well-known binary number systems (i.e., signed magnitude, 1's and 2's complement) offer different advantages and disadvantages, and thus interesting tradeoffs [22]. Redundant number systems [2], [8], and associated encoding variations, enable us to perform digit-parallel addition with a small constant latency. Such redundant number systems may be used as the primary mode of number representation in special application settings or as intermediate or internal forms, with attendant input conversion and output reconversion, for general use.

A suitable encoding of a redundant digit set can further improve the performance of redundant arithmetic operations. For example, a special encoding of double-carry digits in [7] leads to improved latency, area, and power. Other examples of encoding alternatives include representing a symmetric signed digit in the interval $[-\alpha, \alpha]$ as a signed-magnitude, 1's- or 2's-complement digit [11] or as a stored-transfer digit [9]. Encoding of an asymmetric signed digit in $[-2^{h-1}, 2^{h}-1]$ in hybrid-redundant format [24] constitutes yet another example. Various encodings may exhibit different levels of efficiency in implementing the same arithmetic algorithm. Theoretical studies, such as the generalized signed digit number systems [20], may open new horizons for explorations of encodings and implementation techniques that enhance the performance of digital arithmetic operations and systems [1], [12].

The redundancy index ρ of a digit set $[-\alpha, \beta]$ is defined as the difference between the number of digit values (i.e., $\alpha + \beta + 1$) and the radix r of the number system [20]. The higher the redundancy index, the greater the number of bits needed to faithfully represent each digit and often, the longer the potential delays in digit-parallel arithmetic. In many cases, a redundancy index of $\rho = 2$ is adequate, and one never needs to go beyond $\rho = 3$, for carry-free addition in radices higher than 2 [20]. However, proper choice of the redundancy index ρ, coupled with suitable encoding of the resulting digit set, may allow for a faster and/or more compact VLSI realization. Such variations in redundancy indices and associated digit-set encodings are the main focus of this paper. Much of what we present deals, directly or indirectly, with facilitating area-time tradeoffs in the VLSI implementation of arithmetic operations on redundant operands.

Stored-transfer representations [9], [14], weighted bit-set encodings for redundant digit sets [12], and representation paradigms of high-radix signed-digit number systems [11] are all motivated by area-time tradeoff concerns, improvement in the representation coverage, speed of arithmetic, and/or regularity in VLSI implementation. Similarly, hybrid-redundant number systems introduced in [24], extended in [25] and, based on [1], improved with regard to implementation in [17], provide a framework for the efficient design and implementation of digit-parallel addition for a class of redundant number systems. Briefly, a hybrid-redundant number is composed mostly of normal, positively-weighted bits (posibits), with some radix-2 positions holding redundant digits. The hybrid signed-digit (HSD) number systems, as originally described in [24], entailed a composition of nonredundant and redundant positions as an alternative to fully redundant number systems where redundancy appears in every digit position. But later extension of the concept seems to have focused on fully redundant number systems with a variety of redundant digit sets [25]. Unfortunately, the design and implementation of a rather general-purpose redundant arithmetic based on the original notion and subsequent extension of hybrid redundancy engenders some limitations, such as the following, where the last two apply only to the extended form in [25]:

- Considerable difference in the range of positive and negative numbers, leading to inefficiencies in the implementation of subtraction.
- Inapplicability of conventional carry acceleration methods, and the associated highly optimized circuits, due to the use of nonstandard adder cells.
- Inability to faithfully cover, as a representation paradigm, almost all symmetric digit sets as well as many other useful digit sets.
- Lack of representational closure of true hybrid-redundant adders in both the original [25] and subsequently improved [17] efficient adder cells.
- Increasing the likelihood of apparent overflow due to tendency of addition operation to reduce the resultant digit set in some implementations.

To circumvent these problems, which are more fully explained in Sections 2 and 3 of the paper, and exploit the strength of hybrid redundancy in facilitating arithmetic system design with specific area-time tradeoff goals, we reformulate and extend the hybrid redundancy scheme within the framework of weighted bit-set encodings in Section 4. Our quest for more efficient and VLSI-friendly carry-free addition schemes for hybrid-redundant numbers leads us to a scheme for the encoding of negabits (i.e., negatively weighted bits) in Section 5, where we explore different functionalities for standard full-adders, with no added elements (not even inverters), in the summation of any collection of three negabits and posibits. This leads, in Section 6, to new designs for efficient adder cells for both nonredundant and redundant positions in a hybrid-redundant representation. Section 7 shows the power of extended hybrid redundancy scheme in deriving symmetric hybrid-redundant number systems with arbitrary spacing of redundant positions; a property that is vital to exploitation of the main advantage of hybrid redundancy (i.e., spacing of redundant digits to match the area-time design goals). This is followed, in Section 8, by implementation details of an efficient, regular, and representationally closed adder/subtractor for symmetric extended-hybrid-redundant operands and conversion from 2's-complement representation in Section 9. Section 10 concludes the paper.

The extended dot notation used in this paper is described in Table I for ease of reference. Posibit is an ordinary bit with a value in {0, 1}, and its heavy-dot symbol is the same as in standard dot notation commonly used in computer arithmetic. Negabit is a negatively weighted bit, with a value in {–1, 0}. Finally, a "redundant dot" appears in any position containing a value whose range is wider than a posibit or negabit and is usually realized through multiple posibits and/or negabits. For example, a redundant position holding a value in [–1, 2] is realizable by one negabit and two posibits. The latter three bits would appear in a column corresponding to the radix-2 weight of the redundant dot. When the latter substitution is made in the third example of Table I, the resulting dot-notation diagram is said to be "3-deep," meaning that the tallest column of dots contains 3 posibits and/or negabits. According to this terminology, an ordinary unsigned or 2's-complement binary number, appearing in the first two examples of Table I, is 1-deep, and a carry-save or borrow-save number is 2-deep.

Table I. The extended dot notation used throughout this paper.

	Symbol	Name	Meaning	Example of use (8 radix-2 positions)
1	●	Posibit	Value in {0, 1}	● ● ● ● ● ● ● ●
2	○	Negabit	Value in {–1, 0}	○ ● ● ● ● ● ● ●
3	®	Redundant	Value in [$-\nu$, π]	® ● ● ● ® ● ● ●

2. Properties of Ordinary Hybrid Redundancy

Hybrid-redundant (or partially redundant) number systems are weighted radix-2 number systems with some redundant radix-2 positions (holding a digit value from a redundant digit set), but mostly nonredundant positions (holding a posibit). Special practical cases, where the redundant digit sets are limited to those representable by two bits, have been studied in detail [25]. The redundant digit sets for these cases are shown in Table II, where we have also included the notation used in [25], for ease of reference and comparison, and added the last two entries that also meet the restriction for 2-bit representation. Another restriction, as explicated in [25], is that only posibits are allowed in nonredundant positions.

Table II. Redundant radix-2 digit sets that are representable with 2 bits.

	Type of redundant digit	Digit set: $[-\nu, \pi]$	Notation of [25]	Redundancy index ρ
1	Binary signed-digit (BSD)	[–1, 1]	SD	1
2	Stored double borrow (SDB)	[–2, 1]	$\mathrm{SD3}^{\{-\}}$	2
3	Stored borrow or carry (SBC)	[–1, 2]	$\mathrm{SD3}^{\{+\}}$	2
4	Stored carry (SC)	[0, 2]	CS2	1
5	Stored double carry (SDC)	[0, 3]	CS3	2
6	Stored double borrow (SDB)	[–2, 0]	N/A	1
7	Stored triple borrow (STB)	[–3, 0]	N/A	2

To investigate the theoretical properties of hybrid redundancy, we relax the first restriction, namely, that of 2-bit redundant positions. Later, we present our main extension by also relaxing the second restriction via allowing negabits, as well as posibits, in nonredundant positions.

Definition 1 (Posibit hybrid redundancy): A posibit hybrid-redundant number system has k radix-2 positions numbered 0 to $k - 1$, from the least to the most significant position. Each position may be nonredundant, holding a posibit (i.e., a normal bit in [0, 1]), or redundant with a digit in $[-\nu, \pi]$, where $\nu, \pi \geq 0$ and $\nu + \pi > 1$. The digit in radix-2 position i ($0 \leq i < k$), whether nonredundant or redundant, has the weight 2^i. □

We use *posibit hybrid redundancy* to emphasize that in the original notion of hybrid redundancy, nonredundant positions ought to hold a single posibit. At one extreme of every position being redundant and using the same digit set, hybrid redundancy coincides with radix-2 generalized signed-digit (GSD) representations [20]. At the other extreme of no redundant position, a posibit hybrid-redundant number system represents unsigned binary integers. Thus, the claim in [25] that 2's-complement numbers may be considered as a special case of ordinary hybrid redundancy is not valid, given that the negatively weighted sign position of a 2's-complement number violates the requirement for nonredundant positions.

Definition 2 (Periodic hybrid redundancy): A posibit hybrid-redundant number system is periodic if the separation or distance between two consecutive redundant positions remains constant, with the period h being one more than the constant distance. □

Thus, each radix-2^h digit of a periodic hybrid-redundant number system has $h - 1$ nonredundant radix-2 positions and a single redundant radix-2 position. Definition 2 could have been more general (e.g., by allowing multiple redundant positions with varying digit sets in each period), but we use this restricted definition to expose the limitations implied in [24] and [25]. Prior applications of hybrid redundancy (e.g., [19]), mainly in the design of multipliers, have all used periodic subclasses that correspond to radix-2^h digit sets encoded by zero or more posibits followed or preceded by a redundant digit. Such periodic hybrid-redundant number systems can be viewed as efficient encodings for special classes of GSD representations. However, there exist useful GSD number systems, with symmetric digit sets, that cannot be represented via posibit hybrid redundancy. In such cases, we cannot exploit the main benefits of hybrid redundancy, that is, area-time tradeoff. For example, the radix-8 GSD representation with digits in [−5, 5] has no viable representation in posibit hybrid redundancy (see Lemma 1 and Corollary

1 below). We will show later that the subclass of symmetric posibit hybrid-redundant representations is very limited and that efficient implementations, based on the adder cells in [25] and [17], exist only for fully redundant binary signed-digit (BSD) and minimally redundant radix-4 number systems (Corollary 3), both of which had been studied and used prior to, and in contexts other than, hybrid redundancy. We will focus on periodic hybrid redundancy. However, much of what we present pertains to the processing of single radix-2^h digits. Therefore, our results may be applied to nonperiodic cases, for the latter can be studied as a weighted collection of digits with different radices (mixed-radix positional number system).

Definition 3 (Periodic right-, left-, and free-hybrid redundancy): In a hybrid-redundant number system of period h (based on Definition 2), the position index for the redundant radix-2 digit in $[-\nu, \pi]$ may be 0, $h - 1$, or $0 < g < h - 1$ (mod h). We refer to these variants as right-, left-, or free-hybrid redundancy, respectively. Taking each period of the hybrid-redundant representation as a radix-2^h GSD position, the corresponding digit set of a right-hybrid-redundant representation is $[-\nu, 2^h + \pi - 2]$, that of a left-hybrid-redundant representation is $[-2^{h-1}\nu, 2^{h-1}\pi + 2^{h-1} - 1]$, and for a free-hybrid-redundant representation with the redundant digit located in an arbitrary position g is $[-2^g\nu, 2^g(\pi - 1) + 2^h - 1]$. □

Example 1 (Variants of posibit hybrid redundancy): Table III shows examples of left-, right-, and free-hybrid-redundant numbers with three radix-2^h digits in dot notation. The dot notation used is defined in Table I. □

Table III. Variants of posibit hybrid redundancy, with ® in $[-\nu, \pi]$.

Variant	Dot notation	Position g of ®	Radix-16 digit set
Left-hybrid	®●●● ®●●● ®●●●	$3 = h - 1$	$[-8\nu, 8\pi + 7]$
Right-	●●●® ●●●® ●●●®	0	$[-\nu, \pi + 14]$
Free-hybrid	●®●● ●®●● ●®●●	2 $(0 < g < h - 1)$	$[-4\nu, 4\pi + 11]$

Lemma 1 (Symmetry of digit sets associated with periodic hybrid-redundant representations): For periodic radix-r ($r = 2^h > 2$) posibit hybrid-redundant representations with redundant digit in $[-\nu, \pi]$, there is no symmetric digit set for left- or free-hybrid redundancy, while symmetric right-hybrid redundancy is possible for all $h > 1$, provided the radix-r redundant digit set is $[-\alpha, \alpha]$, with $\alpha = \nu = \pi + r - 2$.

Proof: Consider a radix-r ($r = 2^h$) hybrid-redundant digit set $D = [-2^g\nu, 2^g(\pi - 1) + r - 1]$, with the redundant digit being in radix-2 position $0 \le g \le h - 1$. For D to be symmetric as $D = [-\alpha, \alpha]$, we must have $2^g\nu = 2^g(\pi - 1) + 2^h - 1$ or $\nu = \pi - 1 + (2^h - 1) / 2^g$. Obviously, the latter equation has integer solutions for ν and π only if $g = 0$ (i.e., right-hybrid), leading to $\alpha = \nu = \pi + r - 2$. □

Note that for $h = 1$ and $\nu = \pi$, where the left-, right-, and free-hybrid categorization does not apply, the number system is fully redundant and symmetric (e.g., BSD).

Corollary 1: There is no symmetric radix-r ($r = 2^h$) posibit hybrid-redundant number system with the digit set $[-\alpha, \alpha]$ for $\alpha < r - 2$. □

Corollary 2: A symmetric radix-r ($r = 2^h$) right-hybrid-redundant number system with redundant digits in $[-\nu, \pi]$ and $\pi \geq 2$, is over-redundant (i.e., the redundancy index ρ of its radix-r digit set satisfies $\rho \geq r$). Furthermore the minimum redundancy index for such a radix-r symmetric digit set is $\rho = r - 3$ and it occurs when $\pi = 0$. □

Corollary 2 shows that symmetric posibit hybrid redundancy is possible only for highly redundant digit sets satisfying $\rho \geq r - 3$, while, according to the results in [20], $\rho \geq 3$ (2) is always (in most cases) sufficient for carry-free addition, and even $\rho = 1$ allows limited-carry addition, that is, carry-free addition with some look-back (see Definition 4).

In a hybrid-redundant adder, the adder cell of a redundant radix-2 position does not propagate the incoming transfer (e.g., carry or borrow). Transfers generated by redundant or nonredundant positions may ripple up to the next redundant position, where they sink. This process is depicted in Fig. 1, where the larger boxes representing adder cells in redundant positions are intended to reflect the greater complexity of those cells relative to adder cells in nonredundant positions.

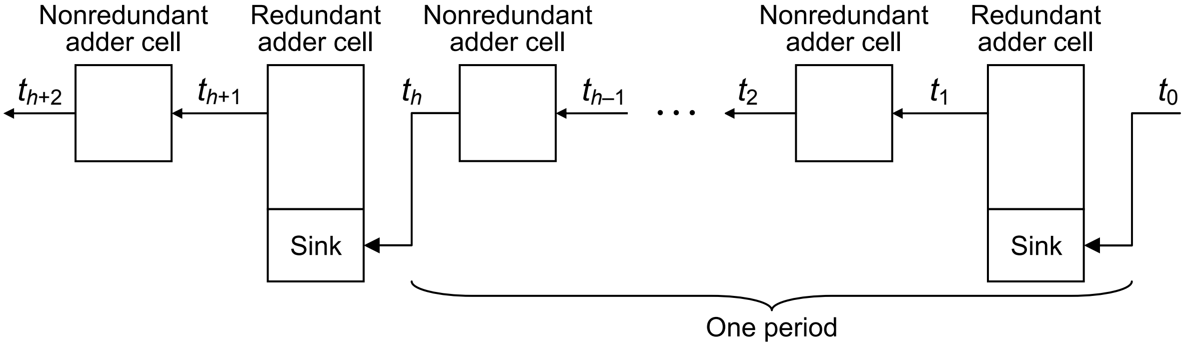


Fig. 1. Schematic representation of an adder for right-hybrid-redundant numbers.

To keep the complexity of adder cells in check, it is desirable to restrict the cardinality of redundant digits to 4, thus making them representable with 2 bits (i.e., the minimum possible for a redundant digit). This constraint leads to 1 bit of redundancy per radix-2^h digit. Unfortunately, such encoding efficiency is gained at the cost of narrowing the spectrum of symmetric hybrid redundancy to only one case besides the fully redundant BSD number system, as is stated below.

Corollary 3 (Restricted symmetry with single redundancy bit): In the case of single redundancy bit per radix-2^h digit, there are only two possible symmetric digit sets in right-hybrid-redundant number system: fully redundant BSD and minimally redundant radix-4.

Proof: Applying the constraint $\nu + \pi \leq 3$ (i.e., 2-bit encoding of redundant digits) to the result of Lemma 1 (i.e., $\nu = \pi + 2^h - 2$) leads to $\pi \leq 5/2 - 2^{h-1}$. Given that $\pi \geq 0$, the latter inequality holds only for $h \leq 2$. The case $h = 1$ leads to $\nu = \pi = 1$ (i.e., fully redundant BSD). The case $h = 2$ results in $\pi = 0$ and $\nu = 2$ (i.e., minimally redundant radix-4). □

In constant-time addition of radix-r redundant numbers, the sum digit in radix-r position i, is a function of the operand digits in the same position i and at least those of position $i - 1$ [20].

Definition 4 (Look-back): The number of consecutive radix-2^h operand digits in the right context of a radix-2^h position i, which contribute to the value of the sum digit in position i, constitutes the look-back of position i. □

For digit sets with ρ ≥ 3 and most cases of ρ = 2 (a few cases of ρ = 2 and all cases of ρ = 1), it has been shown that the required look-back is 1 (2). In other words, the sum digit in radix-2^h position i is a function of four (six) operand digits; the two operand digits in radix-2^h position i and the two in radix-2^h position $i - 1$ (and the two in radix-2^h position $i - 2$) [20]. It is interesting to note that for the minimally redundant case of ρ = 1, the look-back of 2 leads to more complex addition schemes. Thus, the representational cost reflected in ρ ≥ 3 may be more than compensated for by the need for smaller look-back. The few cases of ρ = 2 which require a look-back of 2 are best avoided, because they offer no advantage to compensate for the more complex addition scheme.

Definition 5 (Partial look-back): When, depending on the encoding and implementation (e.g., HSD in [24]), some bit positions of a look-back digit do not contribute to the derivation of a position sum, the look-back is said to be partial. □

The abstract view of a hybrid-redundant adder in Fig. 1 is based on a primary perception of complete separation of adder cells for redundant and nonredundant positions. The only connection between the two kinds of cells would be through carry and/or borrow propagation. But Phatak et al. have used a technique called equal-weight grouping (EWG), which entrusts the higher bits of a digit in radix-2 position $i - 1$, together with lower bits of a radix-2 digit in position i to a single adder cell in position i. This adder cell has been shown to be less complex than one designed without EWG. To investigate the consequences of EWG for hybrid-redundant addition, we consider the 2-bit representation of a redundant digit x_i to be $\langle x^h{}_i\, x^l{}_i \rangle$, with $x^h{}_i$ and $x^l{}_i$ having the weights $\pm 2^{i+1}$ and $\pm 2^i$, respectively. We then define EWG formally as follows.

Definition 6 (Equal-weight grouping, EWG): The higher weighted bit of a redundant digit in radix-2 position $i - 1$ has the same weight as the lower-weighted bit (only bit, in the case of a nonredundant position) of the digit in position i, thus constituting a group of 2 equally weighted bits, regardless of bit polarities. EWG allows us to intermix the processing of bits from various radix-2 positions in order to obtain a more efficient hardware realization. □

Definition 7 (Representationally closed addition): An addition scheme is representationally closed when the two operands are from the same number system (i.e., the equally weighted digits of the two operands belong to the same digit set) and the value of the resultant sum digit, in the corresponding digit-position, also belongs to the same digit set. Furthermore, representational closure requires that these identical digit sets all have the same encoding. □

Representational closure is vital for general-purpose arithmetic, where the same adder circuit is reused to process the results of previous additions. But equal-weight grouping, although simplifying adder cells in fully redundant adders, does not always lead to representational closure in true hybrid-redundant addition. For example, Figs. 2 and 3 represent, by means of digit-set conversion [15], a fully SDB-redundant and a true SDB-hybrid-redundant addition. Composition of two SDB digits results in the interval [–4, 2], whereas that of two posibits produces [0, 2]. The processes of decomposition (e.g., [–4, 2] = 2 × [–2, 0] + [0, 2]) and recomposition (e.g., [–2, 0] + [0, 1] = [–2, 1]) are self-explanatory. As is evident from Figs. 2 and 3, the negabits of operands in position $i - 1$, contribute to the generation of negabits in position i. This causes a representational shift which, owing to the addition of fully redundant operand, remains hidden in Fig. 2, but that is clearly visible in Fig. 3. It is easy to see that the same representationally shifted behavior occurs for true SDC-hybrid-redundant addition with redundant digits in [0, 3], while a direct adaptation of an addition scheme based on the adder cells given in [7] for the same number system would be representationally closed. Also representationally closed addition of true SDB-hybrid operands is certainly possible (e.g., [13]). The sink functionality of position i may be seen in Fig. 3, where carry propagation starts at position $i + 1$.

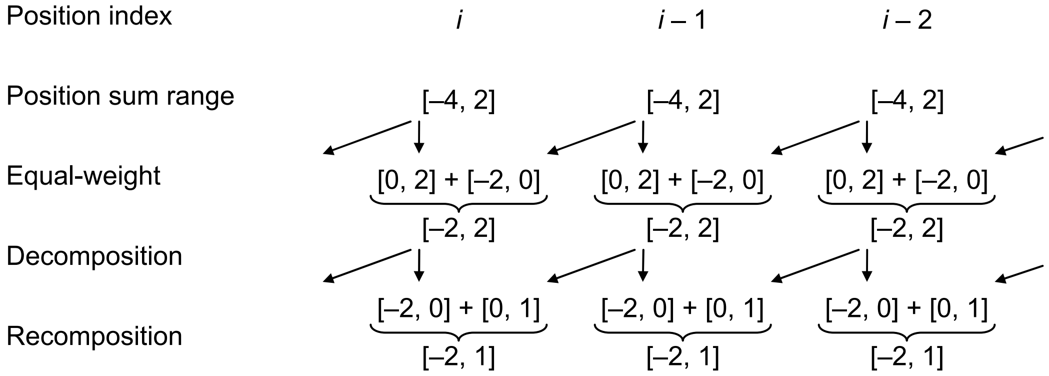


Fig. 2. Addition of fully redundant SDB operands with equal-weight grouping.

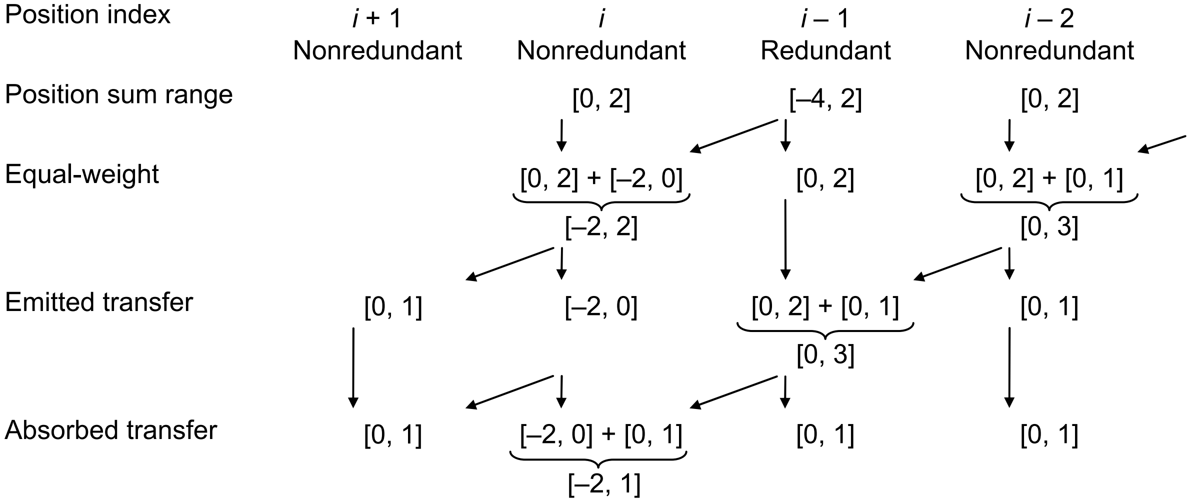


Fig. 3. Addition of true SDB-hybrid-redundant operands with equal-weight grouping.

The fully redundant SDB-hybrid addition of Fig. 2 is based on digit set conversion of [17], where the resultant digit may assume any value of the original digit set [–2, 1]. But an adder cell for the same purpose, offered in [25], does not preserve the operand's digit set and produces digit values in [–1, 1]. A brief assessment of the consequences of this reduction in digit sets is offered below.

Definition 8 (Digit set preservation): The digit set of a number representation is preserved under an arithmetic operation if the result digit may assume all the values in the digit set. □

Example 2 (Impact of digit set nonpreservation): The digit set conversions of Figs. 2 and 3 preserve the digit set [–2, 1]. But the addition scheme of [25] for SDB-hybrid digits reduces the digit set [–2, 1] to [–1, 1], as noted in [17]. Briefly, with the scheme in Figs. 2 and 3, two –1 redundant digits in position *i* are converted to –1 1 (in positions *i* + 1 and *i*) via equal-weight grouping, leading to a digit –2 in position *i* + 1. On the other hand, the addition scheme of [25] decomposes the resulting –2 digit into –1 0, thereby affecting position *i* + 2. □

A drawback of the digit-set nonpreserving addition scheme, mentioned in Example 2, is that addition of most significant digits may signal a false overflow. The digit-set preserving scheme may also signal an apparent overflow [21], but this is less likely.

3. Realization of Hybrid-Redundant Adders

The adder presented by Phatak and Koren for BSD-hybrid-redundant operands (first entry in our Table II, Fig. 1 of [24]), with its redundant radix-2 positions utilizing the adder cell of [18], requires 42 (32) transistors for redundant (nonredundant) positions. Phatak and Koren's corresponding design for redundant radix-2 positions with SDB or SCB redundant digits (second and third entries in our Table II, Fig. 3a of [25]) requires seven multiplexers, a few gates, and several inverters. An analysis of the latter of these two adders shows that the effect of equal-weight grouping is to produce a representationally shifted result in which the redundant position moves from i at the input to $i + 1$ at the output (see also the explanations following Definition 7 in Section 2). The foregoing discussion suggests that for designing a true SDB-hybrid-redundant adder that is representationally closed, specialized adder cells (besides the multiplexer-based design cited above) are needed for isolated redundant positions and immediately higher-weighted nonredundant positions. A high-level design for such adders is offered in [13].

Aoki et al. [1] have shown that an augmented (4; 2)-compressor, with some input/output inverters, can be used for redundant positions of a BSD-hybrid-redundant adder. Kornerup has used such augmented (4; 2)-compressors not only for the redundant position, as above, but also in place of the multiplexer-based cell of Phatak and Koren (see Sections 4.1 and 4.2 in [17]). However, none of these modifications leads to representational closure when applied to a true hybrid-redundant adder. Aoki et al. [1] have also shown that standard full-adders, augmented by suitable input/output inverters, may receive and produce negabits as well as posibits. This obviates the need for special cell designs for nonredundant positions of a hybrid-redundant adder. The inverters in the carry chain will cancel each other out, but the inverters needed for inputs and sum negabits lead to some overhead when compared with unmodified full-adders.

In Section 6, we will show that by inverted encoding of negabits, to be introduced in Section 5, the combining of posibits in a nonredundant position and the incoming borrow or carry can indeed be delegated to a conventional, unmodified full-adder. The benefits of such a design are the use of highly optimized standard full-adder cells (e.g., [3], [4], [26]) and the possibility of carry acceleration within multiple nonredundant positions by means of ordinary binary carry-lookahead circuits (again, readily available in highly optimized forms); neither of these benefits is applicable when realizing hybrid-redundant adders with specialized adder cells.

4. WBS Encodings and Hybrid Redundancy

Weighted bit-set (WBS) encoding of a redundant number system [12] has a fixed number of radix-2 positions, each holding a collection of zero or more equally weighted posibits and negabits. WBS encoding allows the representation of any GSD digit set, including those of hybrid-redundant systems. Furthermore, aperiodic hybrid-redundant number systems, not covered by the GSD paradigm, can also be represented by WBS encoding. For example a posibit-hybrid-redundant number, as in Definition 1, can be represented by a WBS encoding, where nonredundant positions hold a posibit and there is a collection of ν negabits and π posibits in redundant positions, representing $[-\nu, \pi]$. Canonical WBS encodings, where each redundant radix-2 digit set is 3-valued and a proper subset of $[-2, 2]$, are particularly useful for efficient constant time addition. All the variants of posibit hybrid-redundant numbers of Table II may be represented by canonical WBS encoding (see Table IV).

Definition 9 (WBS encoding, redundancy pattern, and canonical encoding): A WBS encoding Ω has k radix-2 positions, where each position i ($0 \le i \le k - 1$) holds π_i (≥ 0) posibits and ν_i (≥ 0) negabits representing the digit set $[-\nu_i, \pi_i]$. The cardinality of Ω equals the value of the (possibly redundant) radix-2 number $M = (m_{k-1}\, m_{k-2} \ldots m_1\, m_0)_2$, where $m_i = \pi_i + \nu_i \ge 0$ is the bit multiplicity of position i. The redundancy pattern of Ω is defined as the possibly redundant radix-2 number $R = (\rho_{k-1}\, \rho_{k-2} \ldots \rho_1\, \rho_0)_2$, where $\rho_i = m_i - 1$. The encoding is canonical if $1 \le m_i \le 2$ (or $0 \le \rho_i \le 1$), for all i, i.e., the digit set in each radix-2 position is [–2, 0], [–1, 0], [–1, 1], [0, 1], or [0, 2], which is representable by two equally weighted negabits, one negabit, a pair of one posibit and one negabit, one posibit, or two posibits, respectively. Given that $m_i \le 2$, a canonical encoding is a 2-deep WBS encoding, unless all $m_i = 1$, where the encoding is 1-deep and nonredundant. □

The unusual option of $m_i = 0$, possibly leading to noncontiguous number systems, has been studied in [12], but is irrelevant to hybrid redundancy. Any k-position posibit hybrid-redundant number system (see Definition 1) may be represented by a k-position WBS encoding, where the latter has a posibit (ν_i negabits and π_i posibits) in position i, corresponding to radix-2 position i of the former with a nonredundant (redundant) digit set [0, 1] ($[-\nu_i, \pi_i]$).

Example 3 (WBS encoding): A 6-deep WBS encoding of a 9-position posibit-hybrid-redundant number system, with three redundant digit sets [–3, 3], [–5, 0], and [–2, 2] from left to right, respectively, is represented at the top of Fig. 5. The overall range of the representable numbers is $[-(3\times2^6 + 5\times2^3 + 2\times2^0),\ 2^8 + 2^7 + 3\times2^6 + 2^5 + 2^4 + 2^2 + 2^1 + 2\times2^0] = [-234, 632]$, with the cardinality $M = 867$. The redundancy pattern is $R = (005004003)_2 = 355 = M - 2^9$ □

An equivalent k-position canonical WBS encoding exists if $M \le 2^{k+1} - 1$ [12], where M is the dynamic range of the source posibit-hybrid-redundant number system. The conversion to a canonical encoding, that is reducing the number of bits in all the k radix-2 positions to at most 2, is possible through the transformations outlined in Fig. 4. See [12] for additional details.

Example 4 (Canonical WBS encoding): Figure 5 depicts the transformation steps for converting the 6-deep, 9-position WBS encoding at the top to an equivalent 2-deep WBS encoding at the bottom. Note that, owing to a singular negabit in position 3, the 2-deep encoding no longer corresponds to a posibit-hybrid-redundant number system. □

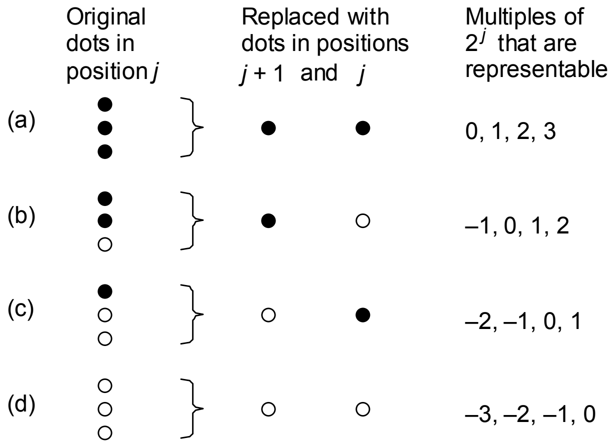

Original dots in position j
Replaced with dots in positions j + 1 and j
Multiples of 2^j that are representable
(a)
0, 1, 2, 3
(b)
–1, 0, 1, 2
(c)
–2, –1, 0, 1
(d)
–3, –2, –1, 0

Fig. 4. Replacement of three equally weighted posibits and negabits.

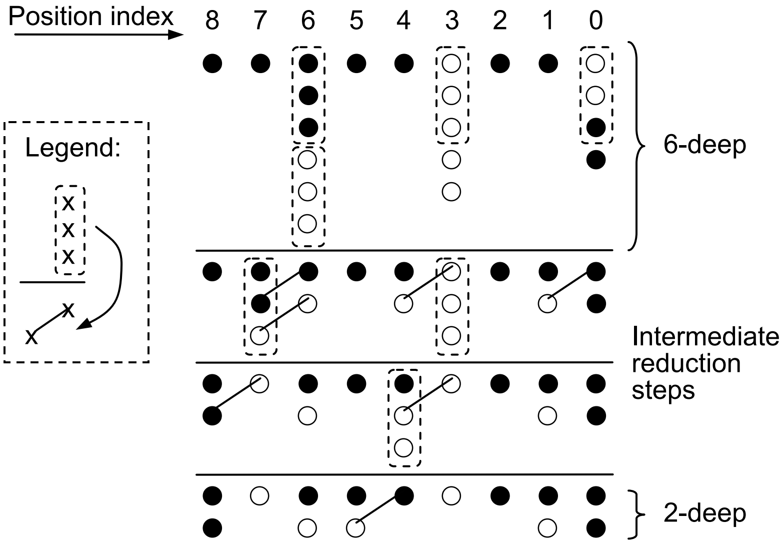


Fig. 5. Transforming a 6-deep encoding into an equivalent 2-deep encoding.

Based on the practical restriction of redundant digit sets to those representable by 2 bits, and in view of the fact that efficient addition schemes exist for redundant number systems represented by 2-deep WBS encodings [13], we are motivated to explore the characteristics of periodic posibit-hybrid-redundant number systems representable by 2-deep WBS encodings.

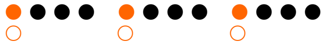
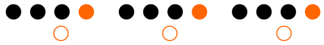
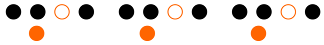
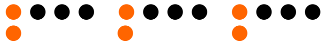
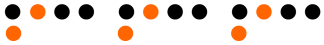
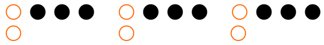
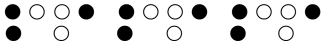
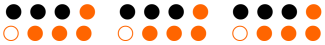
Lemma 2: The digit set of a periodic radix-2^h posibit hybrid-redundant number system Ω with the redundant digit [–ν, π] in position g ($0 \le g \le h - 1$) is representable by a 2-deep h-position WBS encoding iff $\nu + \pi \le 2^{h-g}$.

Proof: The digit set of Ω is $[-2^g\nu, 2^g(\pi - 1) + 2^h - 1]$ (Definition 3) and its cardinality should not exceed the maximum possible cardinality of a 2-deep h-position WBS encoding (i.e., 2^{h+1}). Therefore $2^g(\pi - 1) + 2^h - 1 + 2^g\nu + 1 \le 2^{h+1}$, leading to $\nu + \pi \le 2^{h-g}$. □

Example 5 (Canonical WBS encodings for posibit-hybrid-redundant number systems): Table IV depicts canonical WBS encodings for some radix-16 (h = 4) posibit-hybrid-redundant number systems. The first five entries coincide with those of Table II. Note that in deriving the canonical WBS encoding for the posibit-hybrid-redundant number system of row 7, using the transformations of Fig. 4, the original posibits are not preserved. In all other cases, however, the pale (dark) dots exactly represent the digit set corresponding to the original redundant (nonredundant) positions. □

Table IV shows that all posibit-hybrid-redundant number systems of Table II are representable by 2-deep WBS encodings. These canonical representations may be alternatively regarded as hybrid-redundant number systems with all redundant positions meeting the constraint ν + π = 2 (BSD, SC, or in [–2, 0]). Other hybrid-redundant number systems with redundant digits of wider range (e.g., those in the last two entries of Table IV), when represented by canonical WBS encoding, can be alternatively regarded as having more redundant positions, all with ν + π = 2. Therefore, one can design representationally closed adders for any posibit-hybrid-redundant system, meeting the condition of Lemma 2, based on the adder cells of Fig. 1 in [24]; directly for BSD and SDB hybrid-redundant and posibit nonredundant positions, and designed similarly for other cases of Table II (see Section 6). Note, however, that needing such a wide variety of adder cells is a disadvantage in VLSI design, which favors regularity.

Table IV. Canonical WBS encoding of some posibit-hybrid-redundant number systems.

	Posibit hybrid-redundant number system			WBS encoding with 3 radix-16 digits
	Composition (digit pattern)	ν + π	*g*	
1	1 BSD in [–1, 1], 3 posibits	2	3	
2	1 SDB digit in [–2, 1], 3 posibits	3	0	
3	1 SBC digit in [–1, 2], 3 posibits	3	1	
4	1 SC digit in [0, 2], 3 posibits	2	3	
5	1 SDC digit in [0, 3], 3 posibits	3	2	
6	1 digit in [–2, 0], 3 posibits	2	3	
7	1 digit in [–4, 2] , 3 posibits	6	1	
8	1 digit in [–8, 8] , 3 posibits	16	0	

Posibit hybrid redundancy does not allow single negabits in nonredundant positions. The third entry of Table IV, with a single negabit in its WBS encoding may appear to contradict this claim. However, one must note that in the implementations offered in [25], this single negabit together with a posibit in the next higher position forms an SBC digit in the same (redundant) position as the negabit, and is thus not considered or manipulated by itself as a nonredundant radix-2 digit. Because a negabit represents the nonredundant radix-2 digit set [–1, 0], we are motivated to extend hybrid redundancy to allow for negabits in nonredundant positions. This implies that, in designing the required adder cells, the negabit would be considered by itself and not as part of a redundant digit.

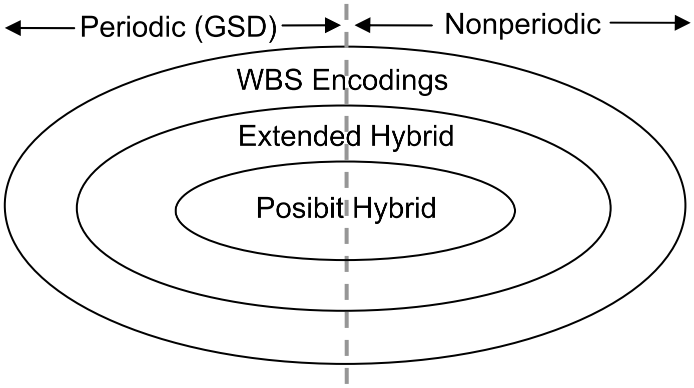


Fig. 6. Relating WBS encodings and their various subclasses.

Definition 10 (Extended hybrid redundancy): A k-position extended-hybrid-redundant number system has k radix-2 positions numbered 0 to $k - 1$ and weighted 2^0 to 2^{k-1}. Each radix-2 position i $(0 \leq i \leq k - 1)$ holds a digit from a digit set $[-\nu_i, \pi_i]$, $\nu_i, \pi_i \geq 0$ and $\nu_i + \pi_i \geq 1$. Position i is redundant (nonredundant) iff $\nu_i + \pi_i \geq 2$ ($\nu_i + \pi_i = 1$). Graphically a redundant position is shown as ®, or by a collection of two or more posibits (●) and negabits (○); a nonredundant position contains exactly one posibit or one negabit. □

Figure 6 depicts the relationships among WBS encodings, GSD number systems, extended-hybrid-redundant number systems, and posibit hybrid redundancy.

5. Inverted Encoding of Negabits

The relative complexity of the adder cells in [24] and [25] is mainly due to carry and borrow propagation within the same circuit. Variants of (4; 2)-compressors and full-adders, augmented with input/output inverters, have been proposed in [1] as efficient tools for the compression and addition of equally-weighted mixed collections of posibits and negabits. Kornerup [17] has used the compressors of [1] as more efficient alternatives to the adder cells of [24] and [25]. Other attempts at similar treatment of equally weighted posibits and negabits (e.g., [6] and [23]) have led to slight variations in full/half-adder circuits for different combinations of posibits and negabits. The difference is often due to extra inverters at inputs and outputs of the standard cells. Although intermediate inverters may cancel each other out in automated VLSI design, inverters for original inputs and final outputs contribute to extra delay, area and power consumption; a problem, that we aim to solve by inverted encoding of negabits. It is well known that inverting all three inputs of a full-adder will result in inverted sum and carry. This hints at using a standard full-adder for adding any three posibits and negabits (see Fig. 4).

Definition 11 (Inverted encoding of negabits): Inverted encoding of negabits is exactly the opposite of the conventional encoding, as used, for example, in the most significant position of standard 2's-complement representation. The lower (higher) value of a negabit, that is, –1 (0), is inversely encoded as 0 (1). We use uppercase (lowercase) letters to designate the logical value of a negabit (posibit). Then the arithmetic value of a negabit X (a posibit x) would be $X - 1$ (x). □

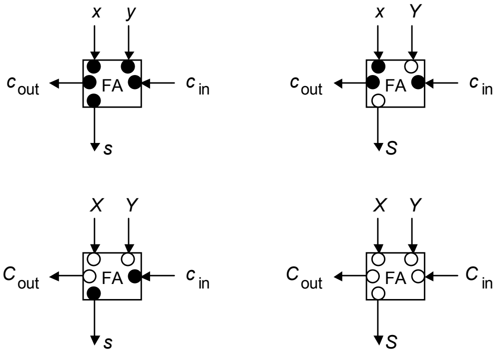


Fig. 7. Universality of a binary full-adder for adding equally weighted posibits (shown as lowercase variables) and negabits (uppercase).

Figure 7 depicts the universal functionality of a standard full-adder as a (3; 2)-counter for any equally weighted collection of 3 posibits and inversely encoded negabits. A full adder with posibit inputs is characterized by the equation $x_1 + x_2 + x_3 = 2c + s$ which relates the arithmetic values of its inputs and outputs. Now, if the posibit input x_1 is replaced by the negabit input X_1, denoting the arithmetic value $X_1 - 1$, the equality $(X_1 - 1) + x_2 + x_3 = 2c + (S - 1)$ shows that the full-adder will produce a negabit sum and posibit carry. The (3; 2)-counter functionality of a full-adder for other combinations of inputs is similarly justified.

Similarly, one could use half-adders to convert any set of 2 equally weighted posibits and negabits to an arithmetically equivalent 1-deep, 2-bit result. This functionality of half-adders is justified by using the equation $x_1 + x_2 = 2c + s$ in the same manner as that of a full-adder in the preceding paragraph. We have shown elsewhere [12] that conventional compressors, independent of how they are implemented, offer a similar functionality in reducing larger collections of posibits and negabits in any combination.

6. VLSI-Friendly Addition Scheme

All the posibit-hybrid-redundant number systems with redundant digits that are representable with 2 bits, and many other extended-hybrid-redundant number systems (discussed in Section 4), can be represented by canonical WBS encodings. Addition of two canonical WBS-encoded numbers is performed by conceptually copying the bits of the 2-deep operands in the bit placeholders of a 4-deep WBS representation. This is then followed by digit-set conversion [15], or reduction to canonical WBS encoding. In fact, if the redundancy patterns (see Definition 9) of operand's encodings are the same, only redundant positions of the operands produce 4-deep results, with nonredundant positions yielding 2-deep results. Otherwise (i.e., operands with different redundancy patterns), a nonredundant position of one operand may align with a redundant position of the other, thus leading to 3-deep positions as well.

With inverted encoding of negabits, reduction of a 4-deep WBS number to a 2-deep one can be delegated to any standard reduction network such as a Wallace tree [27] or Dadda tree [5]. But, the resulting 2-deep number may show an arbitrary redundancy pattern that is not necessarily the same as that of the input operands. This pattern change is what may happen in Algorithm 1 below. Algorithm 2, however, provides for addition results with a preserved redundancy pattern, which, as we will see later, is not necessarily the same as representational closure.

Algorithm 1 (WBS reduction with shifted redundancy pattern)

Input: A 4-deep WBS-encoded number derived by aligning two canonical WBS operands with identical redundancy patterns. See Fig. 8a for an example.

Output: A canonical WBS-encoded result with shifted redundancy pattern, where the redundancy index ρ_{i+1} of the result is equal to ρ_i of the operands, for $i \geq 0$. See Fig. 8c.

I. For each 4-deep position j, use a full-adder to reduce it to a 2-deep position. This leads to a 3-deep position $j + 1$. See Fig. 8b

II. Use a cascade of full-adders for carry-propagate addition starting with a single full-adder at an intermediate 3-deep position $j + 1$ (or position 0), followed by a chain of full-adders for 2-deep positions up to, but not including, the next higher 3-deep position. The carry-out of the full-adder for the leftmost 2-deep position in a chain will stop at the following 3-deep position, where it joins the sum bit generated in that position to form the redundant 2-deep position $j + 1$ of the result. □

The adder cells required for implementation of Algorithm 1 are depicted in Fig. 9. The two full-adders of Fig. 9a, used for redundant positions, may be replaced by any (4; 2) compressor. The single full-adder of Fig. 9b is used for nonredundant positions. Note that a single full-adder for a nonredundant position is the minimum possible.

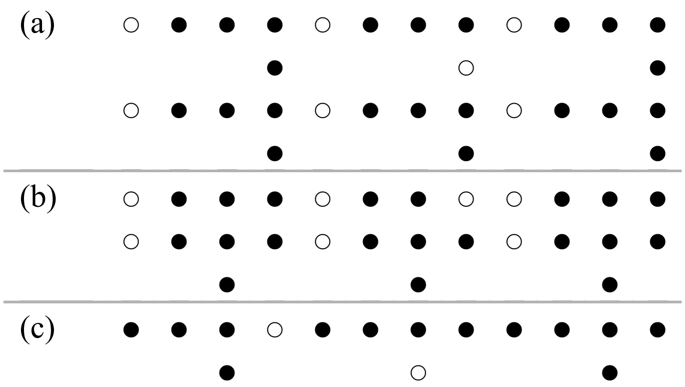
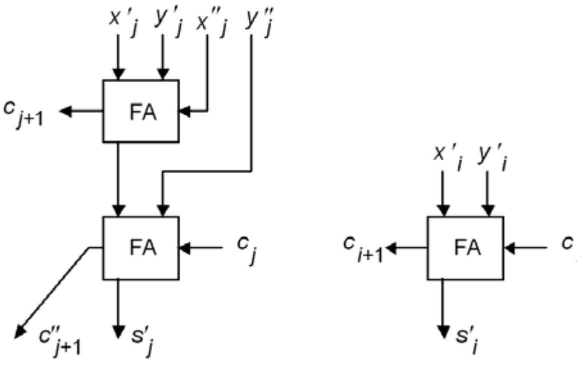


Fig. 8. Reduction with shifted redundancy pattern.



(a) Redundant position (b) Nonredundant position

Fig. 9. Adder cells leading to shifted redundancy pattern.

Algorithm 2 (WBS reduction with preserved redundancy pattern)

Input: A 4-deep WBS-encoded number derived by aligning two canonical WBS operands with identical redundancy patterns. See Fig. 10a, for an example.

Output: A canonical WBS-encoded result with preserved redundancy pattern, where the redundancy index ρ_i of the result is equal to ρ_i of the operands, for $i \geq 0$. See Fig. 10c.

I. Use a full-adder (half-adder) for any 4-deep position j (2-deep position i). This turns each 4-deep position j into a 3-deep position and leaves the multiplicity of 2-deep positions intact (Fig. 10b).

II. Proceed exactly as in step II of Algorithm 1. See Fig. 10c, where the result has the same redundancy pattern as of the operands. □

The required adder cells for Algorithm 2 are depicted in Fig. 11, where the extra cost for preserving the redundancy pattern is seen to be a half-adder per nonredundant position (compare Figs. 9b and 11b). However, the addition latency is the same as that of the circuit for shifted redundancy pattern. Again, the circuit in Fig. 11a may be replaced by a (4; 2)-compressor. Note that the adder cells for redundant positions in both algorithms are identical. Also note that all adder cells are universal in that their functionality does not depend on the polarities of inputs and outputs. This is an important feature of our designs, in the sense of enabling the use of conventional building blocks that, over time, have been highly optimized with regard to complexity, speed, and power requirements.

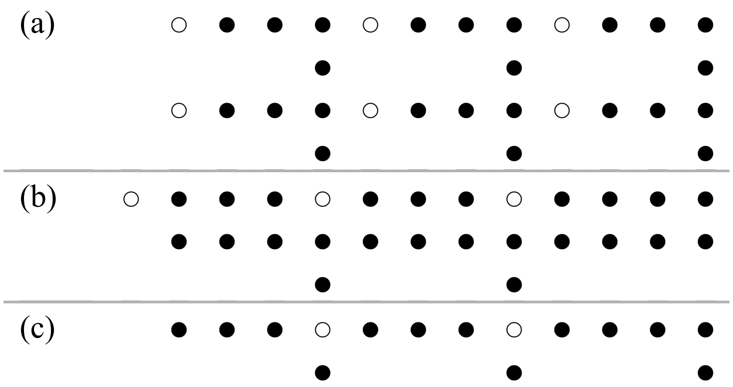


Fig. 10. Reduction with preserved redundancy pattern.

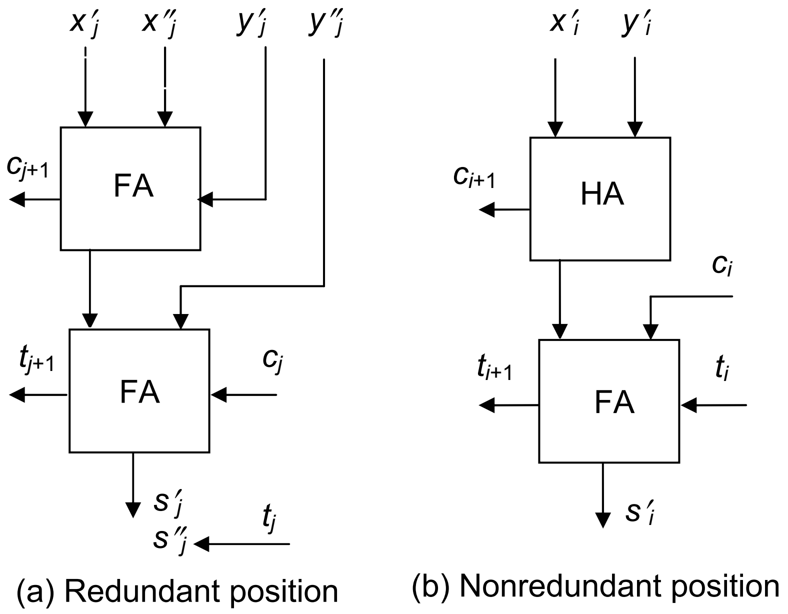


Fig. 11. Reduction cells for preserved redundancy pattern addition.

An approach that preserves the redundancy pattern (e.g., Algorithm 2) does not necessarily lead to representational closure, because the latter requires not only a match in the redundancy patterns of the operands and the result but also identical polarity combinations for like positions. But it is interesting that the adder cells of Fig. 9 preserve the polarity sets of the operands, leading to a representationally shifted result similar to the result of true hybrid redundant adders based on the adder cells in [25] or those in [17]. While the adder provided in [24] for BSD hybrid-redundant operands is representationally closed, neither [25] nor [17] offers or hints at the idea of such an addition scheme for other variants of true hybrid redundancy (i.e., where nonredundant positions do exist).

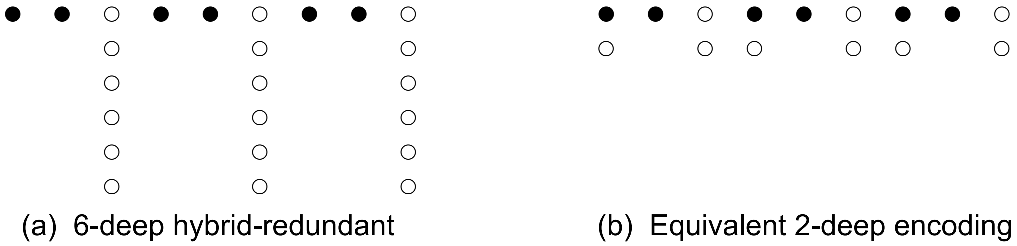
In [12], we have presented a representationally closed adder for SDB-hybrid-redundant operands showing advantages in terms of area, speed, and regularity. In Section 7, we present representationally shifted and representationally closed high-level designs for VLSI-friendly constant-time adders with symmetric extended-hybrid-redundant operands.

7. Symmetric Extended Hybrid Redundancy

Recalling our discussion in Section 2, variants of symmetric posibit hybrid redundancy are limited to hybrid redundant number systems with right-hybrid-redundant digit sets $[-(\pi + 2^h - 2), \pi + 2^h - 2]$ for all $h > 0$, where π is the maximum positive value which can be represented by the right-side redundant position (see Lemma 1). A WBS encoding for such a digit set would have at least $2^h - 2 = 2(2^d - 1)$ negabits in its radix-2 redundant position, where d is the distance between consecutive radix-2 redundant positions. This means that the representation depth in radix-2 redundant positions grows exponentially with the distance parameter d.

The most important characteristic of posibit hybrid redundancy is the design flexibility in allowing an arbitrary number of nonredundant radix-2 positions between radix-2 redundant positions for area-time tradeoff, as it is this number that defines the area requirement and the associated latency for the design. With exponential growth of area for the radix-2 redundant positions when symmetry is a requirement, any attempt to increase h would be ineffective as an area-time tradeoff measure. As an example, for $h = 3$, corresponding to a rather short distance of $d = 2$ between redundant positions, the encoding depth of redundant positions will be $\pi + 6$ (at least 6). Converting such a deep WBS encoding to a 2-deep (canonical) encoding reduces the number of radix-2 nonredundant positions, which is counterproductive as regards to the main advantage of true hybrid redundancy.

Example 6 (Deep symmetric hybrid redundancy): Figure 12 depicts the WBS encoding of a radix-8, 6-deep symmetric hybrid-redundant number system and its equivalent canonical WBS encoding. Each radix-8 digit belongs to [–6, 6]. The reduction process from 6-deep to 2-deep is similar to that shown in Fig. 8. □


(a) 6-deep hybrid-redundant

(b) Equivalent 2-deep encoding

Fig. 12. Depth reduction for a symmetric posibit-hybrid-redundant number system.

Example 6 serves to confirm the result of Corollary 3 that posibit 2-deep hybrid redundancy provides for only two different symmetric digit sets, namely, BSD and the minimally redundant radix-4 digit set [–2, 2]. This observation establishes that posibit 2-deep hybrid-redundant representations are mostly asymmetric, thus essentially denying designers the flexibility of spacing variations to trade off speed for economy (smaller VLSI area) in many cases where symmetry is desired. To reduce the depth of a high-radix symmetric posibit hybrid-redundant representation, it is possible to use more than one radix-2 position for representation of the redundant radix-2 digit set, as was suggested by the equal-weight grouping (see Definition 6).

Example 6 (Shallow encoding of symmetric hybrid redundancy): Consider a 9-position (0 to 8) hybrid-redundant representation with 2 posibits and 8 negabits in positions 0, 3, 6, and a single posibit in every other position (i.e., a 10-deep representation of the radix-8 digit set [–8, 8]). An equivalent 3-deep representation for the above contains a single posibit in positions 1, 4, 7, two posibits in positions 0, 3, 6, and one posibit plus 2 negabits in positions 2, 5, 8. □

The resultant symmetric posibit-hybrid-redundant number system of Example 6 is not a 2-deep WBS encoding; it is thus unsuitable for the efficient universal addition scheme based on the adder cells of Figs. 9 or 11. The process of deriving its equivalent canonical WBS encoding, through the transformations of Fig. 4, leaves a single negabit in each of the originally redundant positions. The canonical WBS encoding thus derived (Fig. 13) no longer represents a posibit-hybrid-redundant number system, but it is an extended-hybrid-redundant number system, as specified by Definition 10. This suggests a general method for constructing a 2-deep WBS encoding to represent a given symmetric range [–α, α]. We begin with a one-position WBS encoding with α posibits, and α negabits, and repeatedly apply the transformations of Fig. 9, until no other similar transformation is applicable [12].

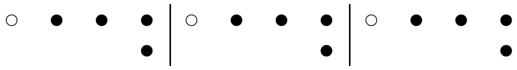


Fig. 13. A canonical WBS encoding of an extended hybrid-redundant number system with the symmetric digit set [–8, 8].

8. Adding Extended-Hybrid-Redundant Numbers

Numbers with arbitrary digit sets can be added digitwise to produce a sum with a digit set whose range is the sum of the ranges of the operand digits. This wider digit set can be kept intact and the result used as an operand in further arithmetic operations. It is also possible to convert the wider digit set to a more convenient one for further processing. Often, however, it is required to obtain results with the same digit set as inputs [16]. Such representationally closed arithmetic is desirable for storage efficiency, reusability of the arithmetic circuits, and regularity in VLSI realization. While encoding-algorithm combinations that are not representationally closed can be useful and are used in practice (e.g., [19]), when a representationally closed scheme is compared against one that is not closed, fairness dictates that the overhead of conversion to the ultimate encoding for the latter be taken into account in any cost/speed comparisons.

Where the two operands in addition are represented with the same canonical WBS encoding, the reduction cells of Fig. 11 may be used to produce a 2-deep result with the same redundancy pattern as that of the operands. Preserving the redundancy pattern is a necessary condition for representational closure, but it is not sufficient; the number of posibits and negabits of the like positions of the result and the operands should be the same as well. One obvious case, in which the latter property is sufficient, is when the encoding consists of only posibits (e.g., SC digit) or only negabits. The adder cells of Fig. 9, however, preserve representational closure, except for a one position left shift in the resultant pattern, that is, the number of posibits and negabits of any position $i + 1$ of the result is equal to that of position i of either operand.

Figure 14 depicts, in dot notation, representationally closed addition of two 3-digit symmetric hybrid-redundant operands with the digit set [–8, 8]. Figure 15 shows a regular adder design for an arbitrary radix-2^h digit i extending from position ih to position $(i + 1)h - 1$, where the only building blocks are full- and half-adders (shaded boxes). Note that cells drawn with dashed lines belong to position $ih - 1$. The addition process is outlined by steps of the algorithm that follows.

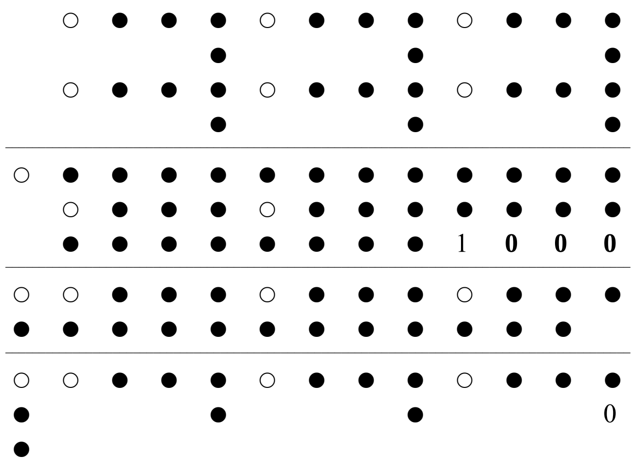


Fig. 14. Representationally closed addition of symmetric hybrid-redundant operands.

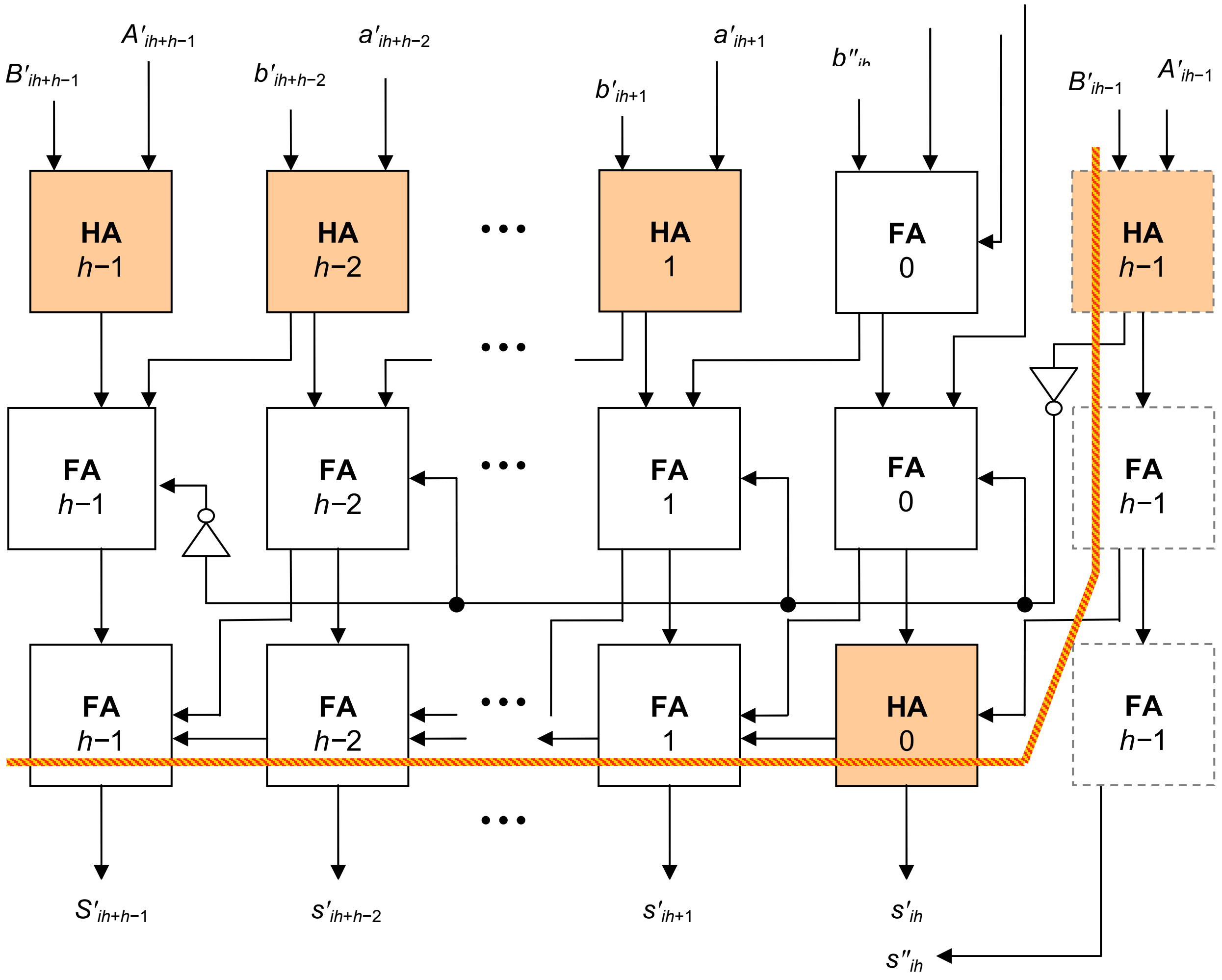


Fig. 15. Representationally closed adder for digit *i* of radix-2^h symmetric hybrid-redundant numbers

Algorithm 3 (Representationally closed addition of symmetric 2-deep extended-hybrid-redundant operands, exemplified by Fig. 13)

1. Replace the 2-deep column of equal-weight negabits by an (h + 1)-position 1-deep 2's-complement number of the same arithmetic value. This produces a new negabit in the next higher negabit position. A standard half-adder can produce the 2-bit 2's-complement sum of 2 negabits. An h-bit sign-extension of the latter produces the desired result; however, due to our inverted encoding of negabits, an inversion is required for sign extension. The required circuitry for this step, a half-adder in the leftmost position of each radix-2^h digit and two inverters, can be seen in Fig. 15.
2. Concurrently with Step 1, use a full-adder (half-adder) in the 4-deep (2-deep) posibit positions to derive a 3-deep intermediate result. Zero-valued posibit and negabit constants (**boldface 0**, regular-face 1) appear in the least significant digit position of Fig. 14 for regularity. The latency for this step is equal to that of one full-adder.
3. Use one full-adder per position to reduce the 3-deep result to one of depth 2. The latency of this step is again equal to that of one full-adder.
4. Use a chain of h full-adders per every h positions to derive the final result. The delay of this step is equal to that of h cascaded full-adders. For large h (say, $h \geq 4$), one may use carry acceleration techniques to ascertain a delay of O(log h). □

The extra cost for subtraction is minimal. We negate the subtrahend by bitwise inversion of each digit, and then perform an addition as above. That simple bitwise inversion of each digit negates that digit, and thus the whole number, is justified as follows.

Theorem 1 (Negating a WBS-encoded symmetric digit): A digit value from a symmetric digit set represented by posibits and inversely encoded negabits is negated by inverting all the bits.

Proof: Let the symmetric digit set, represented over several radix-2 positions, be $[-\alpha, \alpha]$. This implies that the sum of the weights associated with the set of all posibits or all negabits is α. Consider a digit D whose encoding comprises a set of 1-valued posibits of total weight x and a set of 0-valued negabits of total weight y, leading to $D = x - y$, with $x, y \leq \alpha$. The bitwise complemented digit D^{compl} will then have the value $(\alpha - x) - (\alpha - y) = -D$. □

The overall adder circuitry, as depicted in Fig. 15, is comprised of two full-adders and one half-adder per radix-2 position. An inverter per bit and a multiplexer is the minimum possible penalty for subtraction, a bound that is achievable in this case, as noted above. The total addition delay, corresponding to the critical path of Fig. 15 (the heavy broken line) is equal to that of h full-adders and two half-adders. With a carry acceleration circuit, an O(log h) delay can be easily achieved. Note that a representationally shifted adder, based on the adder cells of Fig. 9, has a cost of one (two) full-adder(s) per nonredundant (redundant) position, that is, a total of h + 1 full-adders per radix-2^h digit. The delay, in this case, is equal to that of h + 1 full-adders, almost the same as in the case of representationally closed adder. However, the hardware penalty for representational closure is rather substantial; the equivalent of one extra half-adder (and one extra full-adder) per redundant (nonredundant) position.

9. Conversion from Two's Complement

Conversion from 2's-complement representation to posibit hybrid-redundant representation is quite simple as long as the digit set for redundant positions includes {0, 1}. In particular, this is the case for all representations shown in Table II. Bits are directly transferred from one representation to the other in nonredundant positions (which, by definition, consist of single posibits) and one of the posibit components of a redundant digit is used in redundant positions. The only nonroutine part of the conversion pertains to the sign position. In the case of the last two entries in Table II, corresponding to two entries in Table III which do not include negabits in the leftmost column, conversion of negative numbers is clearly impossible. For the first three entries of Table II, the sign bit of the 2's-complement number can be accommodated in the most significant position of the hybrid-redundant representation. The conversion latency is thus no greater than a single inverter delay.

Based on the preceding discussion, one potential drawback of extended hybrid redundancy is that it may complicate the process of conversion from 2's-complement representation. In this section, we discuss the conversion process in general, showing that any carry propagation in the conversion process can be terminated at a redundant position, and provide an efficient solution for the particular symmetric representation introduced earlier (see Fig. 14).

The general process of conversion from k-bit 2's-complement representation to a given extended-hybrid-redundant representation is as follows. For each posibit in position i of the source number, $0 \le i \le k - 2$, we choose a conversion option from Table V, depending on the bit pattern of the target representation in the same position i. The main objective in this choice is to make the outgoing carry from a redundant position completely independent of the incoming carry for that position. Table V shows how the latter goal can be achieved. Where we have at least two posibits in a redundant position, they can absorb both the source bit and the incoming carry, allowing us to set $c_{\text{out}} = 0$; other posibits (negabits), if any, will be set to 0 (1), corresponding to the arithmetic value 0. For a redundant position with exactly one posibit, we set $c_{\text{out}} = x$, where x is the value of the source posibit, and use xc_{in} (respectively, $x + c_{\text{in}}$) for the target posibit (negabit). Finally, where a redundant position contains no posibit, we choose $c_{\text{out}} = 1$ and set the two target negabits to x and c_{in}. The choices listed for nonredundant positions in Table V are self-explanatory. Because we use conventional or positive carries throughout, carry acceleration techniques with standard circuitry can be easily introduced, if desired.

Table V. Carry propagation rule in conversion from 2's-complement to extended-hybrid-redundant representation.

Source digit x	c_{in}	Nonredundant target (exactly one bit)		Redundant target (at least two bits)		
●		●	○	● ●	● ○	○ ○
0	0	0	1	0 0	0 1	0 0
0	1	1	0	0 1	1 1	0 1
1	0	1	0	1 0	0 0	1 0
1	1	0	1	1 1	0 1	1 1
	c_{out} =	$x\, c_{\text{in}}$	$x + c_{\text{in}}$	**0**	x	**1**

The fact that carry propagation stops at digit boundaries for periodic canonical extended-hybrid-redundant representations is a direct consequence of the fact that for the latter representation to accommodate a continuous interval of integers, each period or *h*-position digit must be able to represent all values in the range $[0, 2^h - 1]$.

In practice, conversion from 2's-complement to extended-hybrid-redundant representation can often be done with no carry propagation and with a latency equivalent to that of a single inverter. For example, in case of the symmetric hybrid-redundant number system depicted in Fig. 13, conversion from 2's-complement representation involves only direct wiring and some inversions, as shown schematically in Fig. 16. Note that the leftmost inverter is needed because of our inverted encoding of negabits.

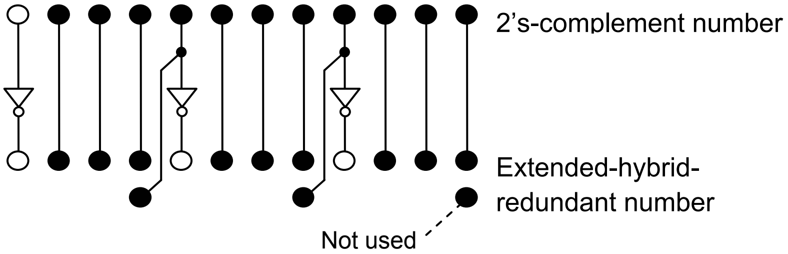


Fig. 16. Schematic view of conversion from 2's-complement to the extended-hybrid-redundant representation of Fig. 13.

10. Conclusions

The hybrid redundancy scheme of [24], extended in [25], constitutes an easily understood concept leading to straightforward management of area-time tradeoffs in the design of hybrid-redundant number systems. The designer has the option of considering as many posibits between the redundant positions as required by cost-performance targets. The redundant positions are practically restricted to at most 4-valued digit sets to enhance the addition speed. The latter constraint, with the help of equal-weight grouping, has led to 2-deep encodings (using the terminology of WBS encodings) of hybrid-redundant number systems. However, the ordinary or posibit hybrid redundancy scheme does not offer the latter design flexibility when shallow symmetric number systems are desired. In such cases, hybrid redundancy fails to provide representational closure in adding true hybrid-redundant operands, does not fully preserve the original digit sets, is incompatible with the direct use of carry acceleration techniques, and lacks support for subtraction by means of the same circuitry used for addition.

In this paper, we provided an in-depth analysis of limitations of posibit hybrid redundancy and showed that these problems can be overcome by two innovations:

- **Allowing single negabits in nonredundant positions.** This possibility, which led to definition of extended hybrid redundancy, helps in designing shallow symmetric hybrid-redundant number systems, which would become impractically deep otherwise (the depth would increase exponentially with the spacing of redundant positions). Furthermore, symmetric digit sets make the negation operation quite efficient and lead to direct reusability of addition circuitry for subtraction. For example, in the case of some common symmetric number systems, negation is performed via bitwise inversion.

- **Encoding negabits in inverted form.** This simple idea leads to the applicability of conventional full/half-adders, counters, and compressors in reducing sets of posibits and negabits and renders carry acceleration techniques directly applicable. For example, a universal (4; 2)-compressor based on inverted encoding of negabits [12] is advantageous, in terms of regularity and use of standard cells, to inverter-augmented (4; 2)-compressor variants proposed in [17] for use in alternate implementations of the adder cells of [25], given that the types and placements of these variants depend on the input/output digit sets. Conventional binary full/half-adders and carry acceleration cells have been studied extensively with regard to area, speed, and energy efficiency [26]; hence, using them in our designs allows a wide choice of predesigned and highly optimized cells.

We showed that when representationally shifted results are acceptable, as is generally the case in true posibit hybrid redundancy with the implementations in [25] and [17], a universal adder may be designed with one (two) full-adders per nonredundant (redundant) position. The adder delay for radix-2^h periodic hybrid-redundant number systems equals that of $h + 1$ full-adders. As shown in the representationally closed adder of Fig. 15, the hardware penalty for the coexistence of symmetry and representational closure, both desired in practice, is the equivalent of one extra half-adder (and one extra full-adder) per redundant (nonredundant) position. Fortunately, however, the addition delay is almost the same (that of h full-adders and two half-adders in series), so the speed penalty is negligible. Conversion from 2's-complement to an extended-hybrid-redundant number system requires limited carry propagation between consecutive redundant positions in the most general case. However, for common symmetric representations, conversion delay reduces to that of a single inverter, which is the minimum possible.

Further research on extended hybrid redundancy schemes may pursue the design of multipliers and dividers as well as efficient circuits for converting from various extended hybrid-redundant formats to 2's-complement binary format. The latter can, of course, be achieved via removal of negabits from all intermediate positions (in a manner similar to step 1 of Algorithm 3) and subsequent use of posibit compression, followed by a carry-propagate addition. However, more efficient schemes may be applicable for specific encodings or classes of encodings.

Acknowledgments

G. Jaberipur's research was supported, in part, by IPM under Grant CS1383-4-02. The authors are grateful to two anonymous reviewers whose comments led to clarification of some points and to an improved presentation.

References

[1] Aoki, T., Y. Sawada, and T. Higuchi, "Signed-Weight Arithmetic and its Application to a Field-Programmable Digital Filter Architecture," *IEICE Trans. Electronics*, Vol. E82-C, No. 9, pp.1687-1698, September1999.

[2] Avizienis, A., "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electronic Computers*, Vol. 10, pp. 389-400, September 1961.

[3] Bui, H. T., Y. Wang, and Y. Jiang, "Design and Analysis of Low-Power 10-Transistor Full Adders Using Novel XOR-XNOR Gates," *IEEE Trans. Circuits and Systems II*, Vol. 49, No. 1, pp. 25-30, January 2002.

[4] Chang, C. H. , J. Gu, and M. Zhang, "A Review of 0.18-μm Full Adder Performances for Tree Structured Arithmetic Circuits," *IEEE Trans. VLSI Systems*, Vol. 13, No. 6, pp. 686-695, June 2005.

[5] Dadda, L., “Some Schemes for Parallel Multipliers,” *Alta Frequenza,* Vol. 34, pp. 349-356, May 1965.

[6] Daumas, M., and D. W. Matula, “Further Reducing the Redundancy of a Notation Over a Minimally Redundant Digit Set,” *J. VLSI Signal Processing*, Vol. 33, pp. 7-18, 2003.

[7] Ercegovac, M. D. and T. Lang, “Effective Coding for Fast Redundant Adders Using the Radix-2 Digit Set {0, 1, 2, 3},” *Proc. Asilomar Conf. Signals, Systems, and Computers*, November 1997, pp. 1163-1167.

[8] Gonzalez, A. F., and P. Mazumder, “Redundant Arithmetic, Algorithms and Implementations,” *Integration: the VLSI Journal*, Vol. 30, pp. 13-53, November 2000.

[9] Jaberipur, G., B. Parhami, and M. Ghodsi, “A Class of Stored-Transfer Representations for Redundant Number Systems,” *Proc. 35th Asilomar Conf. Signals Systems and Computers*, November 2001, pp. 1304-1308.

[10] Jaberipur, G., B. Parhami, and M. Ghodsi, “Weighted Bit-Set Encodings for Redundant Digit Sets: Theory and Applications,” *Proc. 36th Asilomar Conf. Signals Systems and Computers*, November 2002, pp. 1629-1633.

[11] Jaberipur, G., and M. Ghodsi, “High Radix Signed Digit Number Systems: Representation Paradigms,” *Scientia Iranica*, Vol. 10, No. 4, pp. 383-391, October 2003.

[12] Jaberipur, G., B. Parhami, and M. Ghodsi, “Weighted Two-Valued Digit-Set Encodings: Unifying Efficient Hardware Representation Schemes for Redundant Number Systems,” *IEEE Trans. Circuits and Systems I*, Vol. 52, No. 7, pp. 1348, 1357, July 2005.

[13] Jaberipur, G., B. Parhami, and M. Ghodsi, “An Efficient Universal Addition Scheme for all Hybrid-Redundant Representations with Weighted Bit-Set Encoding,” *J. VLSI Signal Processing*, Vol. 42, No. 2, pp. 149-158, February 2006.

[14] Jaberipur, G. and B. Parhami, “Stored-Transfer Representations with Weighted Digit-Set Encodings for Ultrahigh-Speed Arithmetic,” *IET Proc. Circuits, Devices, and Systems*, to appear in 2007.

[15] Kornerup, P., “Digit-Set Conversions: Generalizations and Applications,” *IEEE Trans. Computers*, Vol. 43, No. 5, pp. 622-629, May 1994.

[16] Kornerup, P., “Necessary and Sufficient Conditions for Parallel, Constant Time Conversion and Addition,” *Proc. 14th IEEE Symp. Computer Arithmetic*, April 1999, pp. 152-155.

[17] Kornerup, P., “Reviewing 4-to-2 Adders for Multi-Operand Addition,” *J. VLSI Signal Processing*, Vol. 40, pp. 143-152, 2005.

[18] Kuninobu, S., T. Nishiyama, H. Edamatsu, T. Taniguchi, and N. Takagi, “Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation,” *Proc. 8th IEEE Symp. Computer Arithmetic*, pp. 80-86, 1987.

[19] Lue, J. J. and D. S. Phatak, “Area×Delay (AT) Efficient Multiplier Based on an Intermediate Hybrid Signed-Digit (HSD-1) Representation,” *Proc. 14th IEEE Symp. Computer Arithmetic*, April 1999, pp. 216-224.

[20] Parhami, B., “Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations,” *IEEE Trans. Computers*, Vol. 39, No. 1, pp. 89-98, January 1990.

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

[22] Parhami, B., *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.

[23] Pezaris, S. D., “A 40-ns 17-bit by 17-bit Array Multiplier,” *IEEE Trans. Computers*, Vol. 20, pp. 442-447, April 1971.

[24] Phatak, D. S., and I. Koren, “Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains,” *IEEE Trans. Computers*, Vol. 43, No. 8, pp. 880-891, August 1994.

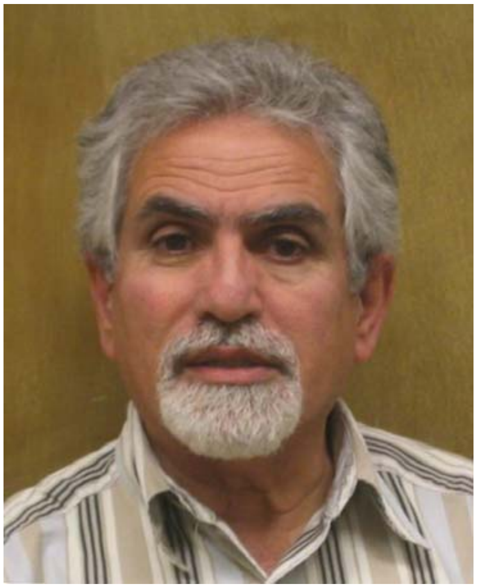
[25] Phatak, D. S. and I. Koren, “Constant-Time Addition and Simultaneous Format Conversion Based on Redundant Binary Representations,” *IEEE Trans. Computers*, Vol. 50, No. 11, pp. 1267-1278, November 2001.

[26] Sayed, A. and H. Al-Asaad, “Survey and Evaluation of Low-Power Full-Adder Cells,” *Proc. Int’l Conf. VLSI*, June 2004, pp. 332-338.

[27] Wallace, C. S., “A Suggestion for a Fast Multiplier,” *IEEE Trans. Electronic Computers,* Vol. 13, pp. 14-17, February 1964.

Authors' Photos and Biographies

G. Jaberipur



B. Parhami

Ghassem Jaberipur received BS in electrical engineering and PhD in computer engineering from Sharif University of Technology in 1974 and 2004, respectively, MS in engineering (majoring in computer hardware) from University of California, Los Angeles, in 1976, and MS in computer science from University of Wisconsin, Madison, in 1979. Since 1979, he has been with the Department of Electrical and Computer Engineering, Shahid Beheshti University, in Tehran, Iran, teaching courses in compiler construction, automata theory, design and implementation of programming languages, and computer arithmetic. Dr. Jaberipur is also affiliated with the School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (IPM), in Tehran, Iran.

Behrooz Parhami (PhD, University of California, Los Angeles, 1973) is Professor of Electrical and Computer Engineering at University of California, Santa Barbara. He has research interests in computer arithmetic, parallel processing, and dependable computing. In his previous position with Sharif University of Technology in Tehran, Iran (1974-88), he was also involved in educational planning, curriculum development, standardization efforts, technology transfer, and various editorial responsibilities, including a five-year term as Editor of Computer Report, a Persian-language computing periodical. His technical publications include over 220 papers in peer-reviewed journals and international conferences, a Persian-language textbook, and an English/Persian glossary of computing terms. Among his publications are three textbooks on parallel processing (Plenum, 1999), computer arithmetic (Oxford, 2000), and computer architecture (Oxford, 2005). He is currently serving on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems* and *International Journal of Parallel, Emergent and Distributed Systems*. Dr. Parhami is a Fellow of both the IEEE and the British Computer Society, a member of the Association for Computing Machinery, and a Distinguished Member of the Informatics Society of Iran for which he served as a founding member and President during 1979-84. He also served as Chairman of IEEE Iran Section (1977-86) and received the IEEE Centennial Medal in 1984.