Journal of VLSI Signal Processing 42, 149–158, 2006

DOI: 10.1007/s11265-005-4177-6

An Efficient Universal Addition Scheme for All Hybrid-Redundant Representations with Weighted Bit-Set Encoding

GHASSEM JABERIPUR
Sharif University of Technology and Shahid Beheshti University, Tehran, 19839-63113, Iran

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

MOHAMMAD GHODSI
Computer Engineering Department, Sharif University of Technology, Tehran 11365, Iran

Received July 09, 2003; Revised September 08, 2004; Accepted December 22, 2004

Abstract. Redundant and hybrid-redundant number representations are used extensively to speed up arithmetic operations within general-purpose and special-purpose digital systems, with the latter (containing both redundant and nonredundant digits) offering cost advantages over fully redundant systems. We use weighted bit-set (WBS) encoding as a paradigm for uniform treatment of five previously studied variants of hybrid-redundant systems. We then extend the class of hybrid-redundant numbers to coincide with the entire set of canonical WBS numbers by allowing an arbitrary nonredundant position, heretofore restricted to ordinary bits (posibits), to hold a negatively weighted bit (negabit). This flexibility leads to interesting and useful symmetric variants of hybrid-redundant representations. We provide a high-level circuit design, based solely on binary full-adders, for a constant-time universal hybrid-redundant adder capable of producing a canonical WBS-encoded sum of two canonical WBS (or extended hybrid) numbers. This is made possible by the use of conventional binary full-adders for reducing any collection of three posibits and negabits, where negabits use an inverted encoding. We compare our adder to previous designs, showing advantages in speed, cost, and regularity. Furthermore we explore representationally closed addition schemes, holding the benefit of greater regularity and reusability, and provide high-level representationally closed designs for all the previously studied variants of hybrid redundancy and for the new symmetric variants introduced here. Finally, we present a new functionality for a conventional (4; 2) compressor in combining any collection of five equally weighted negabits and posibits, and show its utility in the design of multipliers for extended hybrid-redundant numbers.

Keywords: (4;2)-compressor, carry-free addition, computer arithmetic, digit set, redundant number system, signed digit

1. Introduction

Redundant number representations are used extensively to speed up arithmetic operations within both general-purpose and special-purpose digital systems [1, 2]. The speed advantage resulting from carry-free arithmetic with redundant representations is often large enough to offset the format conversion overheads, even

in signal processing and other applications with moderate frequency of arithmetic operations. When the conversion and reconversion circuitry can be shared among multiple function units, redundant representations also lead to savings in VLSI area and power dissipation, thus making them even more attractive. Like conventional digit sets, redundant digit sets can be encoded in any desired way. However, in practice, encodings comprised of weighted positive and negative bits have been found to offer advatages in implementation simplicity and modularity, including the applicability of standard cells used in binary arithmetic [3]. We have thus endeavored to develop a general theory of such representations and the associated arithmetic circuits.

Uniform treatment of negatively weighted and normal (positively weighted) bits is responsible for the simplicity and widespread application of 2's-complement arithmetic [4, 5]. We use *negabits* in $\{-1, 0\}$ for the former and *posibits* in $\{0, 1\}$ for the latter [3]. Negabits have been widely used in redundant number representations. For example, binray signed-digit (BSD) numbers [1] are commonly encoded by using two bits weighted -2^i and 2^i for the position-i digit; viz. the (n, p) encoding [2]. Similarly, in some variants of radix-2 hybrid-redundant numbers [6], redundant digits such as stored-double-borrow (SDB), in $[-2, 1]$, or stored-borrow-or-carry (SBC), in $[-1, 2]$, may be represented by a collection of posibits and negabits, leading to *weighted bit-set* (WBS) encodings [3]. For example, the WBS encoding of a redundant SDB digit consists of two negabits and one posibit in the same position, or, equivalently, of a negabit in position $i + 1$ and a posibit in position i. Other possibly useful variants of digits in redundant positions of a hybrid-redundant number, as enumerated in [6], are stored-carry (SC), in [0, 2], and stored-double-carry (SDC), in [0, 3]. The latter digit set has also been used in the design of redundant adders [7] and for synthesizing certain fast area-efficient multipliers [8].

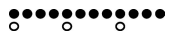
Table 1 depicts symbolic representations for BSD, SDB, SBC, SC, and SDC digits, where a posibit (negabit) appears as • (○). The double-position representations of these redundant digits have been used in Table 2, which depicts five variants of radix-2^h hybrid representations for $h = 4$. The WBS encodings of Table 2 are all 2-deep encodings (i.e., contain no more than 2 dots in any position) with no empty positions; these are known as canonical WBS encodings [3]. The third entry of Table 2 is an example of allowing a negabit in a nonredundant position. By allowing negabits to appear in arbitrary nonredundant positions, canonical WBS encodings, which include all the variants of hybrid redundancy studied by Phatak et al. [6], offer new hybrid-redundant systems not explored before. This nonredundant use of negabits can be seen in 2's-complement numbers and, more recently, in certain stored-transfer representations of redundant numbers [9]. In Section 3, we show that this possibility leads to interesting new symmetric variants of hybrid-redundant digit sets.

Table 1. Single/double-position WBS representations.

Digit	Single-position encoding	Double-position encoding
BSD	● ○	N/A
SDB	○ ○ ●	○ ●
SBC	○ ● ●	● ○
SC	● ●	N/A
SDC	● ● ●	● ●

Addition of two canonical WBS operands is performed by conceptually copying the bits of the 2-deep operands in the bit placeholders of a 4-deep WBS representation. However, with 2-deep operands, it is desirable to convert the sum to a 2-deep encoding as well. In Section 2, we explore an efficient and uniform implementation for constant-time addition of two hybrid-redundant numbers with 2-deep result, where the computed sum need not belong to the same hybrid-redundant number system as the operands (i.e., redundant positions of the result are shifted one position to the left of the redundant position of the operands). This property forces the use of additional hardware for format conversion [6, 10]. We offer representationally closed addition schemes for all the previously studied variants of hybrid-redundant number systems and our new symmetric variants in Section 4. In these im-

Table 2. Five hybrid-redundant number systems.

Composition (digit pattern)	WBS encoding with 3 digits
1 BSD, $h-1$ binary	
1 SDB, $h-1$ binary	
1 SBC, $h-1$ binary	
1 SC, $h-1$ binary	
1 SDC, $h-1$ binary	

plementations, an obtained result belongs to the same number system as the operands.

To multiply two canonical WBS-encoded numbers, we might first derive a partial product bit matrix, composed of posibits and negabits, and then reduce it through compression. In Section 5, we show that inverted encoding of negabits allows us to use standard compressors, such as (3; 2) and (4; 2) counters, for partial product reduction. The number of bitwise products to be dealt with can be 4 times greater than in standard binary multiplicaltion, given the depth of two for each operand. But the second component of each hybrid-redundant operand is relatively sparse compared to the first component. Therefore, one way to reduce the complexity of our multiplier is to reduce the number of positions holding 2 posibits through partial carry assimilation. For example, if 4-bit segments of each 2-deep operand are combined to yield 5-bit binary numbers, with the most significant bit of one number aligned under the least significant bit of the next higher segment, a radix-16 carry-save representation results, for which efficient multiplication circuits have been studied [11].

2. Adding Hybrid-Redundant Numbers

The first step in our addition scheme for WBS encoding of hybrid-redundant numbers is to construct a 4-deep WBS number by simply merging the posibits and negabits in like positions of the two operands, as shown by the vertical alignments of operands in the examples of Table 3. The equal-weight grouping offered in [6] may be considered as a special case. Next we need to reduce the 4-deep result to an equivalent 2-deep result. In the case of SC and SDC hybrid numbers (Table 1),

Table 3. Addition of 2-deep operands with 4-deep results

Composition (digit pattern)	4-deep addition results
1 BSD, $h-1$ binary	
1 SDB, $h-1$ binary	
1 SBC, $h-1$ binary	
1 SC, $h-1$ binary	
1 SDC, $h-1$ binary	

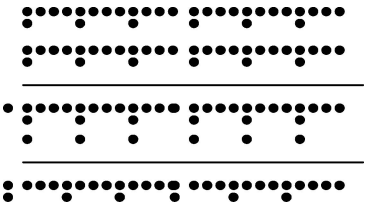


Figure 1. Reduction of the addition result to a 2-deep result.

any conventional reduction scheme may be used for this purpose [12].

For example, one full-adder (FA) per nonredundant position and two FAs in redundant positions are all we need to reduce the 4-deep interim sum of two SC hybrid operands to a 2-deep result (Fig. 1). Note that the sum in Fig. 1 is encoded slightly differently from the operands in that its least-significant group is one position longer (i.e., has $h + 1$ positions). It is easily seen that a reduction scheme similar to that of Fig. 1 is applicable to the addition of SDC hybrid numbers.

The second, third, and fifth rows of Table 1 depict two equivalent encodings for SDB, SBC, and SDC digits, as defined in the second paragraph of Section 1 and summarized in Table 1. The equivalent 3-deep and 1-deep representations for an SDC digit bring to mind the functionallity of a binary full-adder and suggest that similar devices for 3-deep to 1-deep conversions of SDB and SBC digits might be feasible. For example, take the PPM cell used in the design of a borrow-save adder [13], a dual-purpose (rather complex) logic for addition of two SDB or SBC digits [6], and four variants of half-adders that reduce various possible combinations of equally weighted posibits and negabits to carry and sum bits of appropriate kinds [14]. All these seemingly different functionalities can be realized by standard full-adders, provided that we use an inverted encoding for a negabit; that is, encoding -1 as 0 and 0 as 1, which is exactly the opposite of the conventional encoding for negabits.

Table 4 (respectively 5), shows the functionality of a conventional full-adder in reducing a collection of two negabits (posibits) and one posibit (negabit), all in position i, to a negabit (posibit) in position $i + 1$ and a posibit (negabit) in position i. We have used the convention of [3] for variable names: uppercase letters for negabits; lowercase for posibits. The contents of the first three and the last two columns of each table are identical to the truth table for a full-adder, hence the functionality of full-adders for reducing any set of

Table 4. Reduction of two negabits and one posibit.

X_i	Y_i	c_i	$X_i + Y_i + c_i$	C_{i+1}	s_i
0	0	0	−2	0	0
0	0	1	−1	0	1
0	1	0	−1	0	1
0	1	1	0	1	0
1	0	0	−1	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

Table 5. Reduction of two posibits and one negabit.

X_i	y_i	c_i	$X_i + y_i + c_i$	c_{i+1}	S_i
0	0	0	−1	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	2	1	1

three posibits and inversely encoded negabits; the case of three negabits is obvious.

To reduce a 4-deep sum of two hybrid-redundant operands to a 2-deep one, we use one full-adder per nonredundant position and two full-adders for each redundant position. Figures 2a and 2b depict adder cells for redundant and nonredundant positions, respectively. Here, following the convention in [3], primed variables denote the first components of the operands and the result (first row of dots in dot notation), double-primed variables represent bits in the second components (second row of dots), and nonprimed variables indicate intermediate carries. A full-adder in a nonredundant position receives two inputs from the same nonredundant position of the operands and a carry from the previous full-adder, producing a nonredundant sum bit and a carry to the next position (Fig. 2b). In a redundant position, the top full-adder, as in Fig. 2a, reduces three of the bits to a sum bit, feeding the lower full-adder, and a carry to the next higher positioned full-adder. The lower full-adder absorbs the carry from the last position, recieves the sum bit from the top full-adder, and the fourth bit of the redundant position, producing a nonredundant sum bit and a carry that becomes the second bit of the now left-shifted redundant position. These adder cells may be used for all five hybrid-redundant systems of Table 2, which coincide with those covered in [6].

It is interesting to note that in the preceding discussion, the operands need not belong to the same hybrid-redundant system. Moreover, it can be easily verified that they work for addition of any two canonical WBS-encoded numbers. This includes hybrid-redundant numbers with negabits in their nonredundant positions, which we call *extended hybrid-redundant numbers*. However, the result pattern may be slightly different from either operand (i.e., with redundant positions of the result shifted one position to the left of the corresponding redundant positions of the operands). This may not be a problem in intermediate computation steps, where the new format can be simply used as input format for the next computation stage. If the extended dot notation [2] of two 4-deep WBS numbers (possibly resulting from the first step of addition of two canonical WBS operands), irrespective of the polarity of the bits, are identical, then the reduction circuitry is exactly the same.

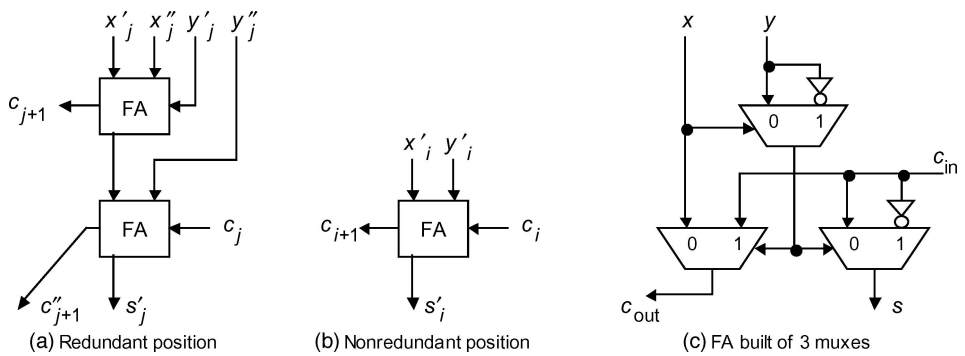


Figure 2. Adder cells for representations of Table 2, and a 3-multiplexer full-adder.

The total adder delay is equal to that of $d + 2$ full-adders, where d is the longest spacing (in terms of the number of nonredundant positions) between two redundant positions. Our universal adder has a number of advantages over previous implementations. The only building block required in our design is full-adder, which leads to more regularity, possibility of using highly optimized FA cells, and employing any standard carry acceleration technique to achieve an O(log d) total delay.

The cost per nonredundant position is minimal (i.e., one FA, as in nonredundant addition), while for redundant positions there is only one extra FA. Given that each FA can be implemeted with three multiplexers (Fig. 2c), our adder cell for redundant positions costs six multiplexers, while the one propsed in [6] for SDB and SBC hybrid cases is made up of seven multiplexers plus a few other gates. This is a pleasant surprise, because the use of standard cells often implies an increase in component count (layout area) or a sacrifice in performance.

3. Symmetric WBS Hybrid Redundancy

Hybrid signed-digit (HSD) representations, introduced in [10] and extended in [6] to allow alternate digit sets in redundant positions, are essentially asymmetric, except for the limiting case that coincides with the fully redundant BSD number system. The reason is that in three of the variants where redundant digits include negative values, there is one equally weighted posibit for each negabit, while other positions hold only posibits. For example, radix-2^h digit sets associated with the hybrid representations shown in Table 2 are $[-2^{h-1}, 2^h - 1]$, $[-2^h, 2^h - 1]$,$[-2^{h-1}, 3 \times 2^{h-1} - 1]$, $[0, 3 \times 2^{h-1} - 1]$, and $[0, 2^{h+1} - 1]$, respectively. We have proved elsewhere [15] that besides the BSD number system, the ordinary hybrid redundancy (i.e., allowing nonredundant positions to hold only posibits) provides for only one other 2-deep symmetric representation, which is the minimally redundant radix-4 signed-digit number system. Figure 3 shows a classification of redundant representations based on weighted bits and, in particular, depicts the place of various hybrid-redundant representations.

A canonical WBS-encoded digit set is redundant if and only if there is at least one position holding a set of more than one posibits and/or negabits [3]. In other words, a position with only one posibit or negabit is nonredundant while any other position is a redundant

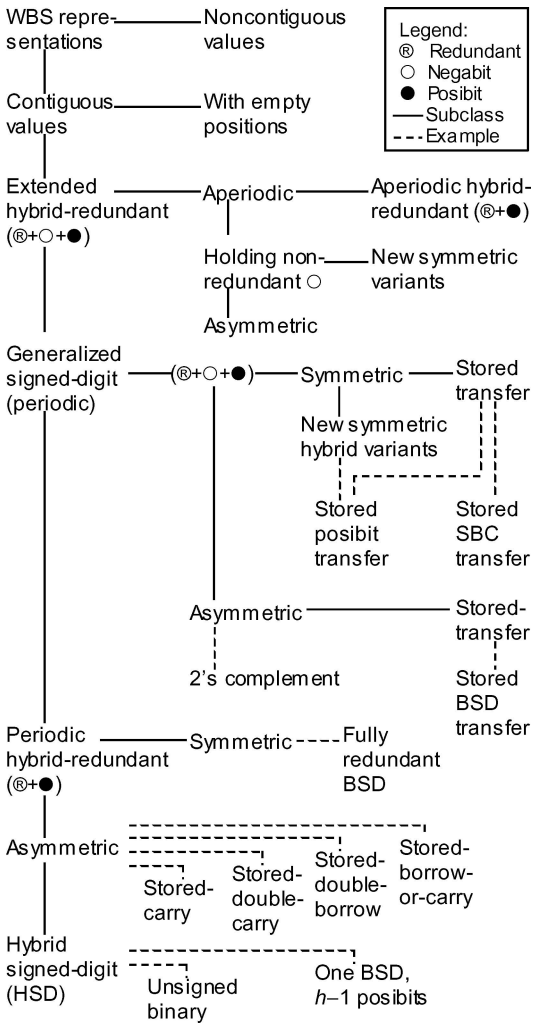


Figure 3 The hierarchy of number representations using weighted components (tree branches go from left to right and top to bottom).

one, given the fact that in a canonical WBS encoding there are no empty positions. This flexibility further extends the hybrid redundancy scheme to allow negabits both in redundant and arbitrary nonredundant positions. We use this extension to design symmetric hybrid-redundant representations with arbitrary different spacing between consecutive redundant positions. For example, consider the periodic radix-16 extended hybrid-redundant number of Fig. 4, where the digit set is $[-8, 8]$. The adder cells as in Figs. 2a and 2b work for this number system as well, but the addition process is not rerpresentationally closed; the pattern of dots in the sum is shifted to the left by one radix-2 position relative to the input operands.

Figure 4. A symmetric hybrid-redundant number system.

4. Representationally Closed Addition

Numbers with arbitrary digit sets can be added digit-wise 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 another, 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 cell designs, and regularity in VLSI circuit implementation.

While encoding-algorithm combinations that are not representationally closed can be useful and are in fact used in practice, when comparing a representationally closed scheme against a scheme that is not closed, fairness dictates that the overhead of conversion from the intermediate representation to the ultimate encoding be taken into account in any cost/speed comparisons. We explore representationally closed constant-time addition schemes for practical cases where the double primed component of the canonical WBS operands are relatively sparse. We present a general addition algorithm below and subsequently apply it to specific cases.

Algorithm 1 (extended hybrid-redundant addition):

Step 1: Add the equally weighted double-primed bits of the second component for the two operands to form a 1-deep sum, possibly left-extended to the next redundant position to preserve sign information.

Step 2: Using one binary full-adder cell per digit position, reduce the 3- or 4-deep WBS number, composed of the two full components of the original operands and the components produced by step 1, to a 2- or 3-deep WBS number. Depth of 4 may occur only in redundant positions.

Step 3: Add the equally weighted digits (where the leftmost position of each digit holds its only redundant binary digit) of the two components of the latter result, in parallel, with special treatment of the redundant positions.

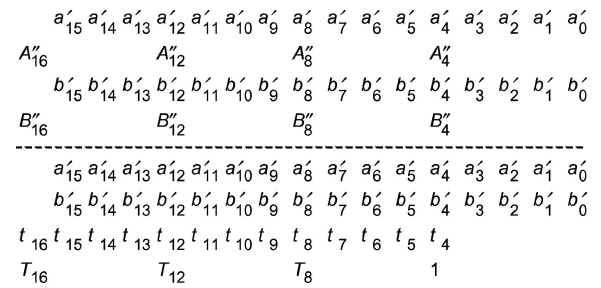


Figure 5. Symbolic representation of step 1 in adding two SDB hybrid-redundant numbers.

We next demonstrate, in detail, the application of Algorithm 1 to addition with SDB hybrid-redundant representation. We also briefly examine the use of this algorithm for other variants. We show that steps 1 and 2 take constant time, whereas step 3, which needs intradigit carry propagation, can be performed in O(log d) time at best, where d is the longest distance between two redundant positions.

Without loss of generality we show the application of Algorithm 1 for radix-2^h periodic SDB hybrid-redundant operands, where each digit includes a full h-posibit primed component, extending from position 0 to $h-1$, and one inverted-negabit double-primed component in position h, overlapping with the least-significant primed posibit component of the next higher digit.

Figure 5 depicts step 1 of Algorithm 1 for 4-digit radix-16 SDB hybrid-redundant operands, where $T_{(i+1)h}, t_{(i+1)h-1} \ldots t_{ih} (i = 1, 2, 3, \text{and } h = 4)$, represent the sign extended 2's-complement sum of two inverted negabits in position ih. For uniformity in treating positions whose indices are multiples of 4, we have placed a 1 in position 4 as the code for an inverted negabit of value 0. Table 6 and Fig. 6 depict the truth table and logic implementation (actually a half-adder) for deriving the 2's-complement sum.

The results of applying step 2 on the 4-deep WBS number of Fig. 5 is shown as the 3-deep WBS number

Table 6. Combining of the double-primed components for SDB hybrid addition.

A''_{ih}	B''_{ih}	Sum	$T_{(i+1)h}$	$t_{(i+1)h-1} \ldots t_{ih+1}$	t_{ih}
0	0	−2	0	1...1	0
0	1	−1	0	1...1	1
1	0	−1	0	1...1	1
1	1	0	1	0...0	0

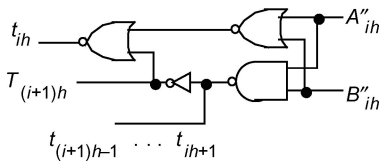


Figure 6. Circuit for reducing the second components of Fig. 5.

in Fig. 7. The first row of full-adders in Fig. 8 constitutes the required hardware, whose operation can start at the same time as that of the circuit of Fig. 6. Step 3 is performed by an $(h-1)$-bit carry-propagate adder in the second row of Fig. 8. The full-adder in position ih receives two posibits and one inverted negabit and generates an inverted negabit sum along with a posibit carry. The posibit carry-out of the full-adder in position $ih-1$ (i.e., s'_{ih} in Fig. 8) is held in position ih and will not propagate beyond there. This bit, together with the inverted negabit sum S''_{ih} of the full-adder in position ih, form the SDB redundant digit of the result in the same position as that of the operands; hence the representational closure property.

The total delay of the adder above is equal to that of $h+1$ full-adders, which is the same as that of our simpler implementation in Section 2, given that $h = d+1$. Note that any carry acceleration method can be applied in a straightforward manner to reduce the delay due to h cascaded FAs within the second row in the design of Fig. 8.

Implementation of an adder for SDB hybrid-redundant numbers is given in [6], where intradigit borrow (as well as carry) propagation and the look-back mechanism complicate the adder cells for nonredundant and redundant positions, respectively. Furthermore, the use of specialized cells makes standard carry acceleration logic inapplicable.

The implementation above works for BSD hybrid numbers as well, for it is the same as SDB hybrid, except that the second component is right shifted by one position. As for the SDC hybrid case, we can use the circuit in Fig. 6 to get a 2-bit sum of the double primed posibits (no extension is needed here). The remaining steps can be followed in Fig. 9. Due to the limited extension in step 1, some positions remain 2-deep. Therefore the corresponding FAs of the first row of Fig. 8 may be replaced by HAs. The SC hybrid representation can be handeled similarly due to its resemblance to SDC hybrid.

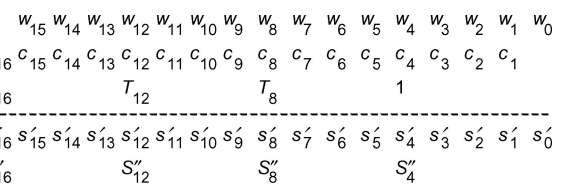


Figure 7. Step 3 of addition for SDB hybrid-redundant numbers.

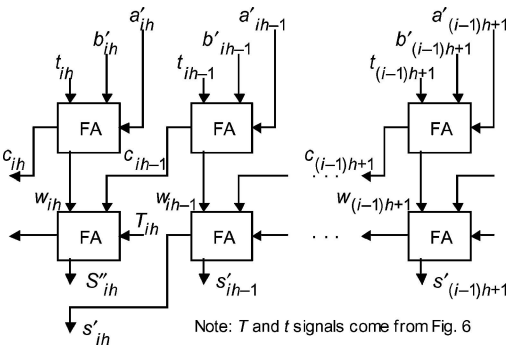


Figure 8. Representationally closed adder for SDB hybrid-redundant numbers.

For SBC hybrid (with double-position redundant digit) and symmetric hybrid numbers, due to existence of negabits in nonredundant positions, step 1 of Algorithm 1 needs to be applied somewhat differently. Figure 10 depicts the situation for symmetric hybrid-redundant numbers, where 0 (1) indicates a posibit (negabit) with constant value 0. In step 1, we make a 1-deep sum of the negabits as well as that of double-primed posibits in redundant positions. Moreover, the reduction to a 2-deep WBS number takes two steps. The generated bits in the leftmost column have been discarded in the final result. A collective nonzero value of those bits indicates an overflow condition.

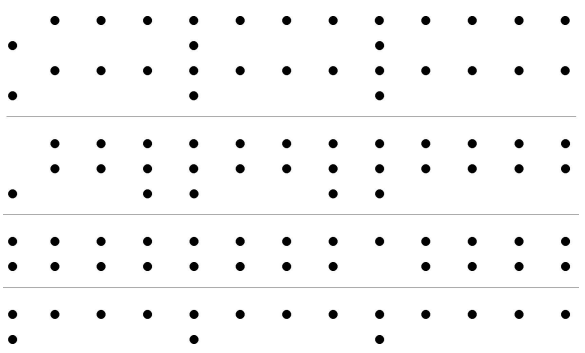


Figure 9. Representationally closed addition for SDC hybrid-redundant operands.

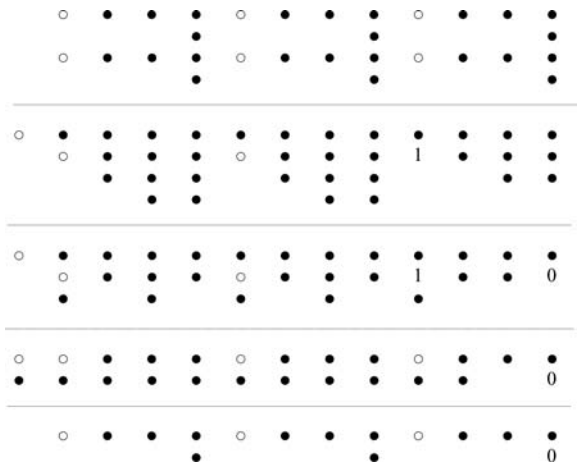


Figure 10. Representationally closed addition of symmetric hybrid-redundant operands.

The same scheme works for SBC hybrid-redundant case, because the encoding is the same, except that the double-primed components have been left-shifted to the next redundant position. The latency is equal to that of $h + 1$ FAs and 1 HA. Given that the circuit of Fig. 6 is actually a half-adder, the complexity of the symmetric hybrid-redundant adder amounts to three FAs per posibit nonredundant position, two FAs plus two HAs per redundant position, and two FAs plus one HA per negabit nonredundant position. Recall that our uniform representationally nonclosed adder of Section 2 had one FA per nonredundant position and two FAs per redundant position. The added complexity is the price paid for symmetry and representational closure. The delay penalty, however, is mimimal, given that the total adder latency is increased only by that of a half-adder.

5. Multiplication of Hybrid-Redundant Numbers

The first step of multiplying two extended hybrid-redundant numbers (or canonical WBS numbers) is to derive the partial product bit-matrix composed of posibits and negabits. Figure 11 depicts the required gates in this step for three possible combinations of posibits and negabits, where upper (lower) case variables indicate negabits (posibits).

We have shown elsewhere [15] that any $(\nu\ ;\ \mu)$-compressor receiving ν equally weighted posibits and negabits in position i produces μ posibits and negabits in positions i through $i + \mu - 1$ such that inputs and outputs have the same collective values. Here we show a similar result for the widely used (4; 2) compressor which recieves 5 equally weighted bits in position i (one of them normally being a carry from position $i - 1$), producing two equally weighted bits in position $i + 1$ and one bit in the same position i (Fig. 12a). The compression process is goverened by the following equation [17]:

$$x'_1 + x'_2 + x'_3 + x'_4 + x'_5 = 2(c' + c'') + s'$$

The arithmatic value $\alpha(X)$ of an inversely encoded negabit X can be expressed in terms of its logical value as $\alpha(X) = X - 1$. Replacing any of the posibits in the equation above by a negabit will add -1 to the left hand side of the equation, which should be compensated for by adding -1 to the right-hand side. The appearance of one, three, or five negabits on the left-hand side, as depicted in Fig. 12, causes the same number of -1s to be added to the right-hand side. These -1s could be absorbed by the sum bit s', and zero, one, or two carry bits, respectively, thus turning to negabits with the same logical values. Two or four negabits on the left-hand side would similarly turn one or two of the carry bits to negabits, respectively. Note that usability of a conventional (4; 2) compressor to reduce any collection of 5 negabits and posibits is independent of the hardware implementation of the compressor.

Any partial product bit-matrix can be reduced to a 2-deep WBS number, by using (4; 2) compressors, and also (3; 2) counters if needed. The resulting 2-deep WBS number can be reduced to a nonredundant 2's-complement number through carry accelaration circuits. It can also be converted to a desired WBS

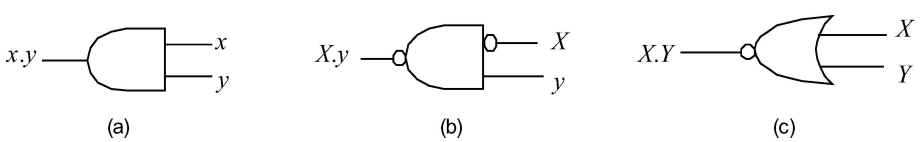


Figure 11. Basic gates for derivation of the partial-product bits.

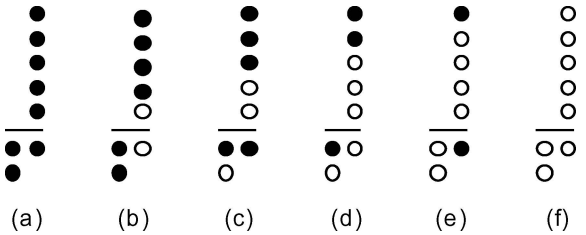


Figure 12. Reduction of alternate collections of 5 negabits and posibits.

encoding (e.g., that of the input operands) through conversion process given in [3].

6. Conclusion

In this paper, we have revisited the previously studied classes of hybrid-redundant numbers by viewing them as subclasses of weighted bit-set (WBS) encodings of redundant representations. We showed that the class of canonical WBS-encoded numbers covers all the variants of the hybrid-redundant numbers previously considered. Moreover, the class of canonical WBS-encoded numbers with a single negabit in some positions represents a new variant of hybrid-redundant numbers where arbitrary nonredundant positions may hold negabits; this is in contrast to standard hybrid redundancy which is restricted to containing only posibits in all nonredundant positions. We noted that this possibility allows for designing new symmetric variants of hybrid-redundant numbers with arbitrary spacing between redundant positions, something not previously accomplished. The new variants, and the flexibility in choosing the encodings for existing systems, allow for optimizations not otherwise possible.

We showed that inverted encoding of negabits leads to the use of a conventional full-adders for the reduction of any set of three equally weighted posibits and negabits to two bits, one with the same weight and the other with double the weight. Using this fact, we provided the high-level design for a universal hybrid-redundant adder capable of adding two extended hybrid-redundant numbers (or canonical WBS numbers) with advantages over previous implementations of hybrid redundancy in terms of circuit regularity, possibility of using standard carry acceleration techniques, shorter critical-path delay, and lower complexity. With regard to the latter, 1 (2) full-adder(s) per nonredundant (redundant) position is required. We further explored representationally closed addition schemes, with the additional advantage of greater reusability, for all variants of hybrid-redundant numbers, including the new symmetric variants introduced in this paper. Finally we showed a new functionality of the popular (4; 2) compressors in reducing any collection of five equally weighted posibits and negabits, and used it in the high-level design of a multiplier for extended hybrid-redundant numbers.

Research on the representational power of WBS encodings, and their various applications in the design of fast arithmetic circuits in signal processing and general-purpose ALUs, is continuing. Problems currently being addressed include refinement of theories for (extended) WBS-encoded representations and deriving design details for the associated multipliers, dividers, and other useful arithmetic operators. Additionally, generalization of concepts and implementation methods to arbitrary two-valued digits (twits), such as those belonging the set $\{-1, 1\}$, have been and are being considered [18].

References

1. A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electronic Computers*, vol. 10, 1961, pp. 389–400.
2. B. Parhami, "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations," *IEEE Trans. Computers*, vol. 39, no. 1, 1990, pp. 89–98.
3. G. Jaberipur, B. Parhami, and M. Ghodsi, "Weighted Bit-Set Encodings for Redundant Digit Sets: Theory and Applications," in *Proc. 36th Asilomar Conf. Signals Systems and Computers*, Nov. 2002, pp. 1629–1633.
4. C.R. Baugh and B.A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Trans. Computers*, vol. 22, 1973, pp. 1045–1047.
5. H. Kobayashi, "A Mutioperand Two's Complement Addition Algorithm," in *Proc. 7th IEEE Symp. Computer Arithmetic*, June 1985, pp. 16–19.
6. D.S. Phatak and I. Koren, "Constant-Time Addition and Simultaneous Format Conversion Based on Redundant Binary Representations," *IEEE Trans. Computers*, vol. 50, no. 11, 2001, pp. 1267–1278.
7. M.D. Ercegovac, "Effective Coding for Fast Redundant Adders Using the Radix-2 Digit Set $\{0, 1, 2, 3\}$," in *Proc. 31st Asilomar Conf. Signals Systems and Computers*, Nov. 1997, pp. 1163–1167.
8. B. Parhami, "Comments on 'High-Speed Area-Efficient Multiplier Design Using Multiple-Valued Current-Mode Circuits'," *IEEE Trans. Computers*, vol. 45, no. 5, 1996, pp. 637–638.

9. G. Jaberipur, B. Parhami, and M. Ghodsi, "A Class of Stored-Transfer Representations for Redundant Number Systems," in *Proc. 35th Asilomar Conf. Signals Systems and Computers*, Nov. 2001, pp. 1304–1308.
10. D.S. Phatak 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, 1994, pp. 880–891.
11. M.I. Ferguson and M.D. Ercegovac, "A Multiplier with Redundant Operands," in *Proc. 33rd Asilomar Conf. Signals Systems and Computers*, Oct. 1999, pp. 1322–1326.
12. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford, 2000.
13. A. Mignotte, J.M. Muller, and O. Peyran, "Synthesis for Mixed Arithmetic," *Design Automation for Embedded Systems*, vol. 5, no. 1, 2000, pp. 29–60.
14. M. Daumas and D.W. Matula, "Further Reducing the Redundancy of a Notation Over a Minimally Redundant Digit Set," *J. VLSI Signal Processing*, vol. 33, 2003, pp. 7–18.
15. G. Jaberipur, "Frameworks for the Exploration and Implementation of Generalized Carry-Free Redundant Number Systems," PhD Dissertation, Sharif Univ. Tech., Tehran, Iran, Dec. 2004.
16. P. Kornerup, "Necessary and Sufficient Conditions for Parallel, Constant Time Conversion and Addition," in *Proc. 14th IEEE Symp. Computer Arithmetic*, April 1999, pp. 152–155.
17. I. Koren, *Computer Arithmetic Algorithms*, 2nd edition, A.K. Peters, 2002.
18. G. Jaberipur, 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: Regular Papers*, vol. 52, no. 7, July 2005, pp. 1348–1357.

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.
jaberipur@sbu.ac.ir

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 200 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). 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.
parhami@ece.ucsb.edu
www.ece.ucsb.edu/Faculty/Parhami

Mohammad Ghodsi Mohammad Ghodsi received BS in electrical engineering from Sharif University of Technology (SUT, Tehran, Iran) in 1975, MS in electrical engineering and computer science from University of California at Berkeley in 1978, and PhD in computer science from the Pennsylvania State University in 1989. He has been affiliated with SUT as a faculty member since 1979. Presently, he is a Professor in SUT's Computer Engineering Department. His research interests include design of efficient algorithms, parallel and systolic algorithms, and computational geometry.
ghodsi@sharif.edu