Weighted Two-Valued Digit-Set Encodings: Unifying Efficient Hardware Representation Schemes for Redundant Number Systems

Ghassem Jaberipur, Behrooz Parhami, *Fellow, IEEE*, and Mohammad Ghodsi

***Abstract*—We introduce the notion of two-valued digit (twit) as a binary variable that can assume one of two different integer values. Posibits, or simply bits, in $\{0, 1\}$ and negabits in $\{-1, 0\}$, commonly used in two's-complement representations and (n, p) encoding of binary signed digits, are special cases of twits. A weighted bit-set (WBS) encoding, which generalizes the two's-complement encoding by allowing one or more posibits and/or negabits in each radix-2 position, has been shown to unify many efficient implementations of redundant number systems. A collection of equally weighted twits, including ones with noncontiguous values (e.g., $\{-1, 1\}$ or $\{0, 2\}$), can lead to wider representation range without the added storage and interconnection costs associated with multivalued digit sets. We present weighted twit-set (WTS) encodings as a generalization of WBS encodings, examine key properties of this new class of encodings, and show that any redundant number system (e.g., generalized signed-digit and hybrid-redundant systems), including those that are based on noncontiguous and/or zero-excluded digit sets, is faithfully representable by WTS encoding. We highlight this broad coverage by a tree chart having WTS representations at its root and various useful redundant representations at its many internal nodes and leaves. We further examine how highly optimized conventional components such as standard full/half-adders and compressors may be used for arithmetic on WTS-encoded operands, thus allowing highly efficient and VLSI-friendly circuit implementations. For example, focusing on the WBS-like subclass of WTS encodings, we describe a twit-based implementation of a particular stored-transfer representation which offers area and speed advantages over other similar designs based on WBS and hybrid-redundant representations.**

***Index Terms*—Arithmetic unit, carry-free addition, computer arithmetic, digit set, hybrid redundancy, number representation, redundant number system, signed-digit number system, stored-transfer representation, weighted bit-set (WBS) encoding.**

I. Introduction

CONTRIBUTIONS to redundant number representation and associated arithmetic systems are of two main types. In abstract studies (e.g., [13], [18], [11]), arithmetic algorithms are presented in terms of digit-level operations, specifying how each result digit is derived from operand digits and auxiliary quantities such as interdigit transfers. Implementation-oriented studies, on the other hand, are often based on specific encodings for the digit sets encountered in solving particular design problems; for example, construction of a high-speed two's-complement full-tree multiplier [22], design of high-throughput floating-point units [14], [17], or enhanced implementation of floating-point addition and rounding [4]. Some contributions of this latter type have dealt with limited classes of digit-set encodings without directly associating them with a specific design problem or application. Examples include the hybrid redundancy scheme [20], [21] and representation paradigms for high-radix signed-digit number systems [7].

This paper aims to fill the gap between the aforementioned contributions. Our focus here is on radices of greater practical interest that are powers of 2. In such radices, a redundant number is formed by a collection of digits, each associated with a power-of-2 weight. Within each digit position, a digit value is also practically encoded as a collection of weighted bits. For example, the possibly asymmetric digit set $[\alpha, \beta]$, with $\alpha \geq -2^{\eta-1}$ and $\beta < 2^{\eta-1}$, might be encoded as an η-bit two's-complement number, giving its bits the weights $-2^{\eta-1}, 2^{\eta-2}, \ldots, 2, 1$. Similarly, binary signed-digit (BSD) numbers [1] are commonly represented by using two bits weighted -2^i and 2^i for the position-i digit, leading to the (n, p) encoding [18]. Also in carry-save [15] and stored-transfer [5] redundant representations, the stored carry or transfer digit is composed of bits with the same weights as those of the main digit. Finally, a hybrid-redundant representation [21] may have redundant positions with stored-double-borrow (SDB) digits in $[-2, 1]$, each of which is encoded using two bits of weight -2^i and one bit of weight 2^i or with a pair of bits of weights -2^{i+1} and 2^i. Under such conditions (i.e., power-of-2 radix and weighted bit-set (WSB) representation of each digit), the number as a whole is encoded by a collection of bits; posibits in $\{0, 1\}$ or negabits in $\{-1, 0\}$, each weighted by a positive or negative power of two, respectively.

The WBS encoding [6] has been studied based on the observation just made. Any addition scheme for WBS-encoded operands entails the problem of combining bits with potentially opposite polarities. Some studies have presented variations of full- and half-adders as a solution to the latter problem. Examples include the **[AU: PLEASE SPELL OUT PPM?]** (PPM) cell, proposed in connection with redundant representations of complex numbers [3] and later used in the design of a borrow-save adder [16], and four half-adder variants that reduce various combinations of equally weighted posibits and negabits [2]. A

Manuscript received November 20, 2003; revised August 4, 2004. This paper was recommended by Associate Editor Y. Wang.

G. Jaberipur is with the Sharif University of Technology and Shahid Beheshti University, Tehran 1983963113, Iran (e-mail: jaberipur@sbu.ac.ir).

B. Parhami is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: parhami@ece.ucsb.edu).

M. Ghodsi is with the Computer Engineering Department, Sharif University of Technology, Tehran 11458889694, Iran (e-mail: ghodsi@sharif.edu).

Digital Object Identifier 10.1109/TCSI.2005.851679

rather complex dual-purpose logic [21] for addition of two SDB or stored-borrow-or-carry (SBC) digits has addressed a similar problem. Inverted encoding of negabits (representing -1 by 0, and 0 by 1, which is exactly the opposite of conventional encoding) allows standard full- and half-adders to be applied for deriving the sum and carry bits of either polarity for any collection of two or three posibits and negabits [9].

Given that *two-valued digit (twit)* sets other than $\{0,1\}$ and $\{-1,0\}$ have found applications in practice (e.g., $\{-1,1\}$ in representing stored-transfer numbers [5]), we are motivated to generalize binary digits to twits and to extend WBS encoding to allow twits in any position. This is taken up in Sections II and III, where we define twits and *weighted twit-set (WTS)* encodings and examine their properties. These include the bias encoding of twits (as a generalization of the aforementioned inverted encoding of negabits), which leads to the possibility of twit manipulations by means of standard full/half-adders. This reliance on the use of standard building blocks makes our results imminently practical. WBS-like encodings, as a subclass of WTS encodings with immediate practical interest, are introduced in Section IV, where we establish necessary and sufficient conditions for contiguity of digit sets and existence of equivalent canonical forms. In WBS-like encodings, each binary position, possibly including noncontiguous twits, represents a contiguous digit set with 0 as a member. WTS interpretation of previously studied redundant number systems (such as generalized signed-digit [18], hybrid-redundant [20], [21], and stored-transfer [5] representations) is taken up in Section 5, where we also provide a general arithmetic framework for WBS-like encoded numbers, based primarily on the notion of digit-set conversion [11], [12], and offer a representationally closed addition/subtraction high level design, for a subclass of WTS encodings. Section VI concludes the paper and offers a comprehensive hierarchical classification of all redundant number representations that the authors have encountered in the literature as instances of WTS encodings.

Various properties of twits and of WTS encodings cited in this paper are stated as theorems and associated corollaries, and supported by formal proofs, in Appendix A. It is our hope that this separation of formal proofs from concise descriptions of representational properties and arithmetic relationships will facilitate the adoption of these ideas by circuit/logic designers. Appendix B contains a list of symbols, abbreviations, and key terms used throughout this paper for ease of reference. Fig. 13 in Section VI, which relates a wide variety of number representations based on WTS encoding, is also provided for reference. Not all elements in Fig. 13 make sense at this point, before introducing a number of concepts in later sections.

II. Two-Valued Digits

Besides negabits and posibits used in the WBS definition [6], other twits, such as transfer digits in $\{-1,1\}$, have been found useful in practice [5]. Also, one could think of an SDC, or stored-double-carry [19], digit in [0, 3] as being represented, with improved encoding efficiency, by a pair of equally weighted twits in $\{0,1\}$ and $\{0,2\}$, respectively, instead of by three equally weighted posibits. Digit sets not including 0, such as [1], [3], cannot be faithfully represented by any collection of posibits and/or negabits. However a posibit and a twit in $\{1,2\}$, both of the same weight, can represent [1], [3] precisely. This motivates us to generalize binary digits to twits in Definition 1 and to extend WBS encoding to include any twit (see Definition 3 at the beginning of Section III).

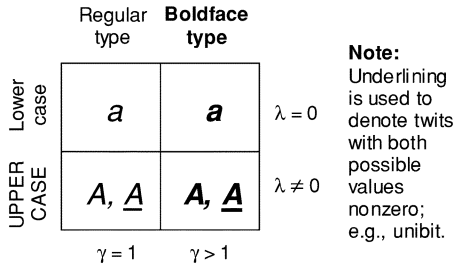


Fig. 1. Conventions for twit symbolic names.

	Name	Lower value	Upper value	Gap size	Dot notation	Symbolic notation
(a)	Bit or posibit	0	1	1	●	x', y'', z'''
(b)	Negabit	−1	0	1	○	X', Y'', Z'''
(c)	Unibit	−1	1	2	▣	$\underline{X}', \underline{Y}'', \underline{Z}'''$
(d)	Doublebit	0	2	2	■	$\boldsymbol{x', y'', z'''}$
(e)	Negadoublebit	−2	0	2	□	$\boldsymbol{X', Y'', Z'''}$

Fig. 2. Some examples of twits or twits.

Definition 1 (Two-Valued Digit): A *twit* has two possible values, λ and $\lambda+\gamma$. A twit encoded as a bit x represents the value $\lambda+\gamma x$, with λ (*lower value*) and $\gamma>0$ (*gap size*) being the twit parameters. If $\gamma=1$ $(\gamma>1)$, the twit is *contiguous* (*noncontiguous*). If $\lambda\neq 0$ $(\lambda>0)$, the twit is *biased* (*unbiased*). The least (highest) representable value by a collection of m equally weighted twits is $\Lambda_{(m)}$ $(\Lambda_{(m)}+\Gamma_{(m)})$, where $\Lambda_{(m)}=\sum_{0\leq i<m}\lambda_i$ and $\Gamma_{(m)}=\sum_{0\leq i<m}\gamma_i$. □

For notational convenience, we use letters to denote twits according to the following conventions. Regular (**boldface**) type is used to denote contiguous (noncontiguous) twits, while lower (UPPER) case is used for unbiased (biased) twits having $\lambda=0$ $(\lambda\neq 0)$. When 0 is not one of the two twit values, we underline the twit's symbolic name. Twits in the same digit position are distinguished by using prime, double-prime, triple-prime, and so on. For ease of reference, these conventions are illustrated in Fig. 1, and some special twits, along with their representations in dot and symbolic notations, are depicted in Fig. 2.

It has been shown elsewhere [9] that a standard full-adder is capable of correct (3; 2) reduction of posibits and negabits, provided that negabits are inversely encoded, with the lower -1 value of a negabit encoded as 0 and the upper 0 value encoded as 1. In other words, the arithmetic value of a negabit with the same logical value as a posibit is biased by -1. This observation may be generalized as follows.

Definition 2 (Bias Encoding of Twits): Encoding of the lower value λ and higher value $\lambda+\gamma$ of a twit as 0 and 1, respectively, is called *bias encoding* (the lower value λ is biased relative to lower value of a posibit). *Twit bias* is then synonymous with the lower value λ. □

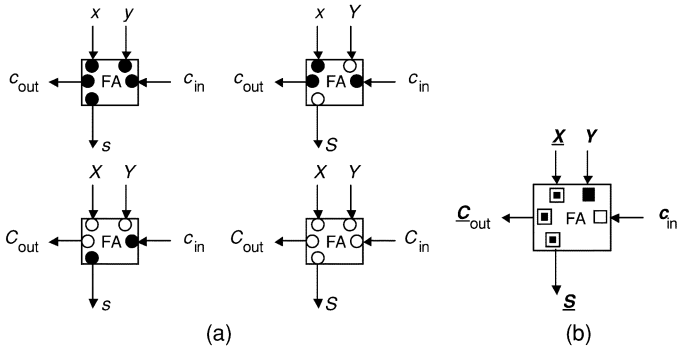


Fig. 3. Twit-FA used for adding various collections of three twits.

Twit Property 1 (Twit-FA): The sum and carry outputs of a standard full-adder, receiving three bias encoded twits with equal gaps, can represent arithmetically correct sum and carry twits with the same gaps. This property, which stems from our special bias encoding of twits, is justified by Theorem 1 in Appendix A. □

The bias λ for a negabit is -1; that is, for a bias-encoded negabit, logical 0 means -1 and logical 1 means 0, which is the opposite of the convention used for the negabit in the most significant position of a two's-complement number. Similarly, for a bias-encoded unibit, logical 0 means -1 and logical 1 means 1; again the opposite of the universally adopted sign convention. However, given that each of the two possible encodings of a twit is the logical inverse of the other, any required conversion is trivial.

Example 1 (Twit-FA): Fig. 3(a) shows the functionality of a standard full-adder as twit-FA for different collections of three posibits and negabits. The functionality of the full-adder of Fig. 3(b) in adding a unibit $(-1+2\underline{\boldsymbol{X}})$, a doublebit $(0+2\boldsymbol{c_{in}})$, and a negadoublebit $(-2+2\boldsymbol{Y})$ is justified by the equation

$$-3+2(\underline{\boldsymbol{X}}+\boldsymbol{Y}+\boldsymbol{c_{in}})=2(-1+2\underline{\boldsymbol{C}}_{\mathbf{out}})+(-1+2\underline{\boldsymbol{S}})$$

where $2\underline{\boldsymbol{C}}_{\mathbf{out}}+\underline{\boldsymbol{S}}=\underline{\boldsymbol{X}}+\boldsymbol{Y}+\boldsymbol{c_{in}}$ represents the normal full-adder functionality. □

Twit Property 2 (Twit Compressor): A standard compressor, normally implemented by a collection of standard full-adders, may receive equigap twits in lieu of input posibits and produce twits with the same gap where one normally sees output posibits. This property is justified by Corollary 1 in Appendix A. □

Twit Property 3 (In-Place Reduction of Twits): Three equally weighted, equigap, bias-encoded twits may be reduced by a full-adder to two equally weighted twits, one with a doubled gap (the carry output) and one with the original gap (the sum output). Furthermore, two equally weighted twits can be replaced by two other twits with the same weights, where the bias of one is increased by a constant and that of the other is decreased by the same constant. This property is justified by Corollary 2 in Appendix A. □

Example 2 (Twit Reductions): An equally weighted collection of two posibits and one negabit may be reduced to a doubly weighted posibit and one negabit with the same weight (per twit property 1), a doublebit and a negabit, a $\{1,2\}$ twit and a negadoublebit (per the first part of twit property 3), or a unibit and a posibit (per the second part of twit Property 3). The last three

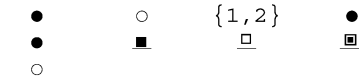


Fig. 4. Two posibits and one negabit, along with three possible in-place reductions.

in-place reductions are depicted in Fig. 4. We will make good use of the latter reduction in generating the unibit transfer of a stored transfer addition in Section V. □

Twit Property 4 (Gaps in Representation): Consider for each twit in a collection of equally weighted twits, the difference between its gap and sum of the gaps of all twits with smaller gap sizes. The largest of these differences equals the maximum distance between consecutive integer values in the ordered collection of integers representable by the twit collection. When the largest difference is 1, the twit collection can represent a contiguous interval of integers. Furthermore, if the representable contiguous interval is $[-\alpha,\beta]$ and includes 0, it may equivalently be represented by an equally weighted collection of α negabits and β posibits. Justifications are provided by Theorem 2 and Corollaries 3 and 4 in Appendix A. □

Example 3 (Representational Efficiency of Twits): The set of three twits $\{0,1\},\{-2,0\}$, and $\{1,5\}$, with gaps of $\gamma_0=1,\gamma_1=2$, and $\gamma_2=4$ meets the conditions of twit Property 4. The set represents integers in $[-1,6]$, which can equivalently be represented by a collection of 1 negabit and 6 posibits. This example demonstrates the representational power of twit collections in enhancing the overall encoding efficiency of redundant binary digits (3 twits versus 7 negabits/posibits). This observation is generalized as twit Property 5 below. □

Twit Property 5 (Size of Twit Representation): A contiguous interval $[\alpha,\beta]$ of integers is representable by the minimum number $m=\lceil\log_2(\beta-\alpha+1)\rceil$ of equally weighted twits whose gaps are $1,2,4,\ldots,2^{m-2},\beta-\alpha+1-2^{m-1}$. For $\beta-\alpha+1=2^m$, encoding efficiency of the resulting representation (see Definition 4 in Section III) is maximal. This property is justified by Theorem 3 and Corollary 5 in Appendix A. □

III. Weighted Twit-Set Encodings

Having defined twits and examined some of their properties, we proceed to introduce a very general twit-based encoding scheme as a tool for unifying, evaluating, and comparing redundant number representations.

Definition 3 (WTS-Encoded Numbers): A k-position WTS encoding is characterized by k integers $m_{k-1},\ldots,m_1,m_0$, where the representation has k radix-2 positions indexed 0 to $k-1$ and the multiplicity of digit position i $(0\le i<k)$ of weight 2^i is m_i (i.e., it is comprised of m_i twits). The twits in position i are numbered 0 to m_i-1, with the jth one in the set $(0\le j<m_i)$ represented as $\{\lambda_{ij},\lambda_{ij}+\gamma_{ij}\}$. We postulate for the most significant position that $m_{k-1}>0$. Other positions may be empty, that is, $m_i\ge 0$ for $0\le i<k-1$. □

Note that WBS encodings [6], elaborated upon in Section IV, constitute special cases of WTS encodings, where the twits are restricted to posibits and/or negabits.

Definition 4 (Characteristics of WTS Encodings): The lowest (highest) value collectively representable by the twits

in position i is Λ_i $(\Lambda_i + \Gamma_i)$, where $\Lambda_i = \sum_{0 \le j < m_i} \lambda_{ij}$ and $\Gamma_i = \sum_{0 \le j < m_i} \gamma_{ij}$. *Positional bias* is a synonym for the lowest positional value Λ_i. The maximum distance for position i (see twit Property 4) is denoted as $d_i^{\max}$. The *effective gap* of the twit $\{\lambda, \lambda + \gamma\}$ in position i is $\varepsilon = 2^i\gamma$. The lowest (highest) value collectively representable by the i rightmost positions of the WTS encoding is Λ_i^+ $(\Lambda_i^+ + \Gamma_i^+)$, where $\Lambda_i^+ = \sum_{0 \le j < i} 2^j \Lambda_j$ is the *partial encoding bias* and $\Gamma_i^+ = \sum_{0 \le j < i} 2^j \Gamma_j$. The lowest (highest) value representable by such a k-position encoding as a whole is Λ^+ $(\Lambda^+ + \Gamma^+)$, where Λ^+ is the *total encoding bias*. The redundancy index of position i is defined as $\rho_i = \Gamma_i - 1$, where a negative ρ_i of -1 occurs in empty positions (denoted by ∇ in our extended dot notation) and $\rho_{k-1} \ge 0$ by Definition 3. The ordered collection $\rho_{k-1} \ldots \rho_1 \rho_0$ of the k positional redundancy indexes is the *redundancy pattern* and $R = \Gamma^+ + 1 - 2^k$ is the *total redundancy index*, which may be represented as the possibly redundant radix-2 number $(\rho_{k-1} \ldots \rho_1 \rho_0)_{\text{two}}$. Similarly, the ith *partial redundancy index* is defined as $R_i = (\rho_{i-1} \ldots \rho_1 \rho_0)_{\text{two}} = \Gamma_i^+ + 1 - 2^i$, with $R_0 = 0$. The total encoding cost is $E = \sum_{0 \le i < k} m_i$, leading to the encoding efficiency $e = \lceil \log_2(\Gamma^+ + 1) \rceil / E = \lceil \log_2(2^k + R) \rceil / E$. □

Definition 5 (Strongly Contiguous WTS Encoding): A *strongly contiguous* WTS encoding is one where each digit position represents a nonempty interval of integers (see twit Property 4) and, consequently, so does the entire encoding. □

Definition 6 (Equivalent WTS Encodings): WTS encodings representing precisely the same set of integer values are *equivalent*. *Strongly equivalent* WTS encodings are equivalent and equiwidth (have the same number k of positions). □

Definition 7 (Complementary WTS Encodings): If the negation of every integer representable by a WTS encoding is representable by another WTS encoding, and vice versa, the two encodings are *complementary*. If each twit of a given WTS encoding is replaced by an inverted twit (e.g., posibits by negabits, negabits by posibits, and doublebits by negadoublebits), with possible swapping of placements in the same position, the encoding that results is *strongly complementary* to the original one. □

Equivalent or complementary WTS encodings that are equiwidth have the same total redundancy indices, but their redundancy patterns may be different in general; redundancy patterns are the same in case of strong complementation. Complementary equiwidth WTS encodings are not necessarily strongly complementary.

Example 4 (Equivalent WTS Encodings): The 8-position WBS encoding (a) in Fig. 5 is equivalent to the 7-position encoding (b). Furthermore, encoding (a) is strongly equivalent to the 8-position encoding (c), strongly complementary to the 8-position encoding (d), and complementary to the 8-position encoding (e). □

IV. WBS-Like Encodings

The digit sets encountered in practice are, almost always, contiguous, and include 0 as a member. These contiguous zero-included digit sets may be represented by a collection of equally weighted posibits and negabits in a straightforward manner, leading to a WBS encoding. However, a noncontiguous

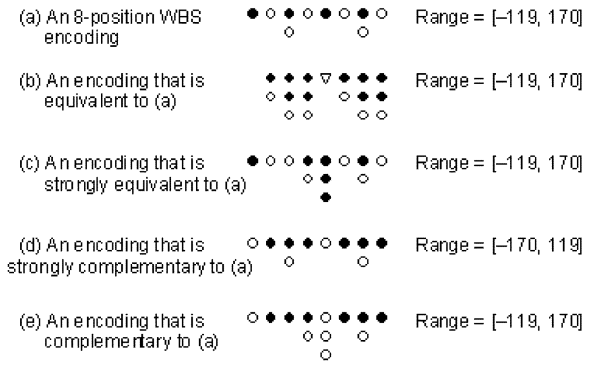


Fig. 5. Equivalent and complementary WTS encodings.

TABLE I
SOME WBS AND EQUIVALENT WBS-LIKE ENCODINGS

Encoding name	WBS encoding	WBS-like encoding	Range
SDB hybrid			–272, 255
Stored transfer			–153, 136
Not named			–119, 170

and/or zero-excluded twit may contribute in the representation of the same digit set and enhance the encoding efficiency (see Example 2). The equivalence of the two representations (i.e., with and without twits other than posibits and negabits) hints that any result obtained for WBS encodings [6] might be valid for WTS encodings with contiguous zero-included digit sets in each nonempty position. Therefore we define the class of WBS-like encodings and review the properties of WBS encodings that are applicable to WBS-like encodings.

Definition 8 (WBS-Like Encoding): A *WBS-like* encoding is a strongly contiguous WTS encoding that meets the conditions of the last part of twit Property 4 in every digit position; that is, each digit position represents an interval of integers including 0 or, equivalently, is representable by a collection of equally weighted posibits and negabits. □

Example 5 (WBS-Like Encodings): Table I depicts some WBS encodings along with their equivalent WBS-like encodings illustrating the advantage of noncontiguous twits in improving the encoding efficiency. The first entry represents a two-digit radix-16 periodic hybrid redundant number system [21] with SDB redundant positions using digits in $[-2, 1]$. The second one is a stored transfer representation [5] with transfer digits in $[-1, 1]$. The third one is a made-up example intended to illustrate the generality of our encodings in that they need not be regular or periodic. □

WBS Property 1 (Contiguity): A WBS encoding is said to be contiguous iff the set of integers represented by the encoding exactly coincides with a contiguous interval of integers. Obviously, A WBS encoding with no empty position is contiguous. But if the right context of an empty position is deep enough to compensate for the missing range caused by the empty position, then the whole encoding could still be contiguous. Formal

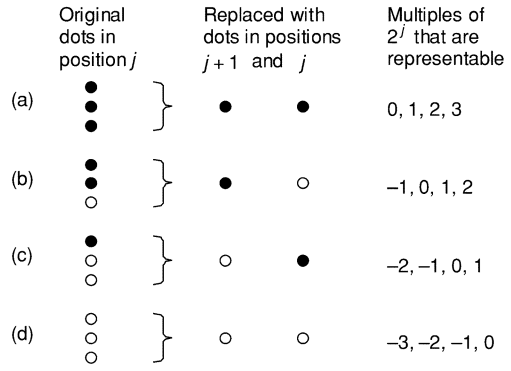


Fig. 6. Substitutions used in the proof of WBS Property 2 (Theorem 5 in Appendix A).

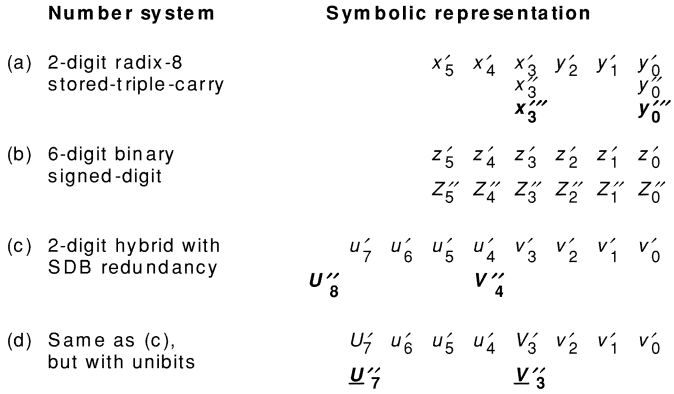


Fig. 7. Symbolic representation of periodic WTS-encoded numbers.

description of this property is provided by Theorem 4 in Appendix A. □

WBS Property 1 suggests that even though it is possible to avoid having any posibit or negabit in a particular position j, doing so would require additional bits in less significant positions (two in position $j-1$, four in position $j-2$, and so on). Thus, for encoding efficiency, it is advantageous to enforce $m_i > 0$ for all i. On the other hand, replacement of a pair of bits of the same polarity in position j by one bit in position $j+1$, through the substitutions outlined in Fig. 6, keeps $m_i \leq 2$, and further improves encoding efficiency. These observations lead us to define the class of canonical WBS encodings.

Definition 9 (Canonical WBS Encodings): A k-position WBS encoding is *canonical* iff it is strongly contiguous (Definition 5) and has $\rho_i \leq 1$ (i.e., $1 \leq m_i \leq 2$) for $0 \leq i \leq k-2$.□

Several strongly equivalent canonical encodings may exist for a given WBS encoding Ω. For example, if Ω is symmetric, any strongly equivalent canonical encoding Ω' leads to another strongly equivalent encoding Ω'' which is strongly complementary to Ω'. Interestingly, these encodings have the same redundancy pattern.

WBS Property 2 (Uniqueness of Redundancy Patterns Among Strongly Equivalent Canonical Encodings): For all equivalent canonical WBS encodings with the same number of positions, the numbers of bits in the like positions are the same. Theorem 5 in Appendix A provides justification for this property. □

WBS Property 2, which is established through the transformations depicted in Fig. 6, has led to the possibility of designing a universal adder circuit for all such encodings [9].

WBS Property 3 (Redundancy of a WBS Encoding): A given k-position WBS encoding is *redundant* iff in any of its strongly equivalent canonical forms, $\rho_j > 0$ for some $j < k$. □

WBS Property 3 is a direct consequence of WBS Property 2. We have deliberately associated the redundancy of a WBS encoding with the redundancy of its strongly equivalent canonical forms because existence of a redundant position by itself does not imply a positive total redundancy index. For example, the 3-position WBS encoding having the redundancy pattern 0–1 2 (i.e., with position 1 empty) is nonredundant, even though its position 0 is redundant.

WBS Property 4 (Efficiency of Canonical WBS Encodings): The encoding efficiency of a canonical encoding Ω is maximal among all WBS encodings strongly equivalent to Ω. This property is established by Theorem 6 in Appendix A. □

V. Arithmetic on WTS-Encoded Operands

While arbitrary WTS encodings can be envisaged and used, circuit implementation of arithmetic functions in VLSI favors regularity in the numbers and kinds of twits associated with the various positions. Thus, we define the class of periodic WTS encodings.

Definition 10 (Periodic WTS Encodings): A k-position WTS encoding is deemed *periodic* iff there exists $h < k$, such that the twit collection of position $i + jh$ is precisely the same as that of position i, for $0 \leq i \leq h-1$ and $0 < j \leq \lceil k/h \rceil - 1$; the smallest such h is the *period*. □

Assuming k to be a multiple of h, a periodic WBS-like-encoded number represents a generalized signed-digit (GSD) number system in radix 2^h utilizing the digit set $[\alpha, \beta]$, with $\alpha = \Lambda_h^+$, and $\beta = \Lambda_h^+ + \Gamma_h^+$, where $\Lambda_h^+ = (\Lambda_{h-1} \ldots \Lambda_1 \Lambda_0)_{\text{two}}$ and $\Gamma_h^+ = (\Gamma_{h-1} \ldots \Gamma_1 \Gamma_0)_{\text{two}}$.

Example 6 (Symbolic Representation of WTS Encodings): The symbolic representations for a 2-digit radix-8 stored-triple-carry (STC) number, a 6-digit BSD number, and a 2-digit radix-16 stored double-borrow (SDB) hybrid-redundant number are depicted in Fig. 7. Note that the digit sets for these WTS-encoded GSD number systems are [0, 10], $[-1, 1]$, $[-16, 15]$, and $[-16, 15]$, respectively. □

A general framework for arithmetic operations with WTS-encoded operands may be established following the general framework of arithmetic for WBS encodings [6]. Given that the addition operation may be viewed as a special case of digit-set conversion [12], and arithmetic functions on WTS operands can always be reduced to one or more addition operations, the central problem in WTS arithmetic is recognized as conversion of a deep digit set to a one with less depth. This is where bias encoding of twits helps in using standard compressors for reducing the representation depth to a desired level. When the input operands and the derived results have the same WTS encodings, the arithmetic is said to be representationally closed. A key example of representationally closed arithmetic, and its associated advantages, has been explained elsewhere [9] in connection with SDB hybrid-redundant operands.

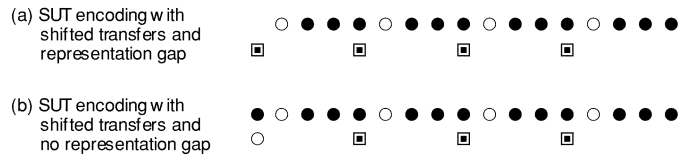


Fig. 8. Stored-unibit-transfer encodings with shifted transfers.

To illustrate the advantages of WTS encoding and of the use of twit-FA's in enhancing encoding efficiency and regularity of VLSI design as well as addition speed, we adapt the representationally closed WBS addition algorithm of [9] to a WBS-like encoded stored-transfer representation. One such encoding can be seen in the second entry of Table I with unibit transfers. In the following, we shift each of the unibit transfer digits h positions to the left as depicted in Fig. 8(a) in order to achieve a wider representation range. Because the most significant position then holds a single unibit, it violates the WBS-like restriction on positional contiguity (Definition 8) and also results in a representation gap by Theorem 4 (see Appendix A). A simple fix is to replace the single unibit in the most-significant position by a posibit and a negabit [Fig. 8(b)]. With these modifications, the representation remains periodic, except for the most-significant digit whose transfer is now a shifted binary signed digit instead of a shifted unibit.

Definition 11 (SUT Representation): The digit set Δ of a radix-2^h periodic stored-unibit-transfer (SUT) representation with shifted transfers is composed of a radix-2^h main part $\Delta' = [-2^{h-1}, 2^{h-1}-1]$ in two's-complement form and a twit transfer part $G = \{-2^h, 2^h\}$, except in the most significant position where the transfer set is $\{-2^h, 0, 2^h\}$. □

It is interesting to note that, due to use of noncontiguous twits (i.e., unibits) in the SUT definition, Δ is not contiguous, but with the most significant position modified to hold a contiguous digit set, the number system as a whole is contiguous. Also, the SUT representation may be regarded as an extended hybrid-redundant number system [10] with the digit set $[-1, 2]$ in redundant positions, but it has no equivalent in ordinary hybrid-redundant scheme [21]. The reason is that the redundant digit set composed of a negabit and a doubly weighted unibit, exactly representing $\{-3, -2, 1, 2\}$, is not contiguous.

We now proceed to provide the high-level design for an adder for SUT operands. Fig. 9 depicts a symbolic representation of addition steps for two 4-digit radix-16 SUT operands, where $T_{(i+1)h-1}$ and $t_{(i+1)h-2} \ldots t_{ih}$ $(i = 1, 2, 3, h = 4)$, denote sign-extended two's-complement sum of two unibits in position ih. The required circuit, actually very similar to a half-adder per each redundant position, is shown in Fig. 10, with a justification provided in Table II. The overall SUT adder is depicted in Fig. 11, where the adder cells in the first row serve as reduction units. The full-adders in the second row are serially interconnected and perform a standard h-bit ripple-carry addition. This part of the circuit can be replaced by any desired fast adder design incorporating carry acceleration. A second-row full-adder in position ih (except in the most significant digit position) generates a sum bit and a unibit transfer by in-place reduction of two posibits and one negabit (see twit Property 3, and Corollary 2 in Appendix A). Position kh needs a different treatment (that is

Fig. 9. Representationally closed SUT addition.

TABLE II
COMBINING OF UNIBIT TRANSFERS FOR SUT ADDITION

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

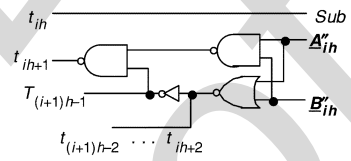


Fig. 10. Circuit for reducing unibit transfers of Fig. 9.

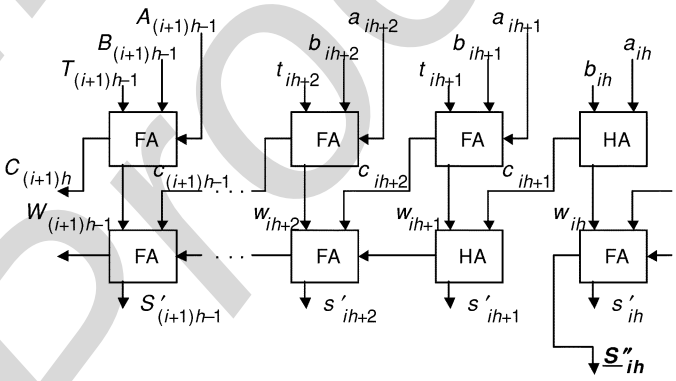


Fig. 11. SUT radix-2^h redundant adder.

why the details for that position are left out in Fig. 9); there are 3 negabits and three posibits to be added: $a'_{kh}, b'_{kh}, A''_{kh}, B''_{kh}$, and transfers coming from the first (C_{kh}) and second rows of full-adders. Fig. 12 depicts the required hardware in position kh, along with overflow detection logic, where the overflow bits do not always indicate a real overflow [8]. This condition of "apparent overflow" is pretty much the norm in redundant number representation schemes [18].

For subtraction, one usually negates the subtrahend, and then performs addition. However, we can apply a more efficient approach based on negating each digit independently. An SUT digit has a two's-complement number as the main part, which

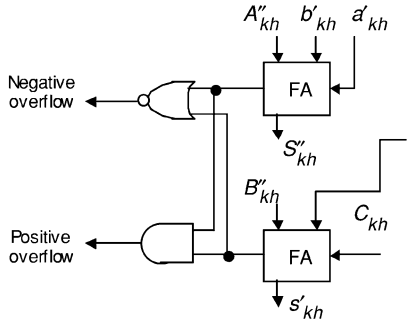


Fig. 12. The cell at the most significant position of our SUT adder.

can be negated via two's complementation, and a unibit transfer which is negated by inversion. To negate the main digit, we simply invert its bits and let $t_{ih} = \mathrm{sub}$, where sub (Fig. 10) is the subtraction control signal (1 for subtraction, 0 for addition). The hardware modification to accommodate subtraction is minimal and consists of replacing the half-adder in the first row of position ih by a full adder. The most significant transfer being a BSD, again needs special treatment: to negate it, we simply invert its posibit and negabit components and do not apply the sub control as in other positions. Based on the description above, the time penalty for negation is minimal and consists of a single inverter delay.

VI. Conclusion

In this paper, we introduced the use of general twits, or twits, that include posibits and negabits as special cases. We showed that WTS encodings cover all positional redundant number systems that have appeared in the literature, including those employing subranges of integers (perhaps excluding zero) and noncontiguous digit sets. Fig. 13 presents a hierarchical classification of all redundant representations that can be obtained from WTS encodings at the root. We showed how bias encoding of twits, as a generalization of inverted encoding of negabits, leads to new functionalities for standard full/half-adders and compressors in reducing equally weighted, equigap twits. The latter possibility led to use of standard reduction (e.g., Wallace tree) and carry acceleration techniques to implement arithmetic on WTS-encoded operands. Focusing on a subclass of WTS representations, those that possess equivalent WBS encodings, a twit-based representationally closed adder design for SUT representation was described. This twit-based design offers advantages over a similar WBS-based implementation of SDB hybrid redundancy [9] in speed and time/logic penalty for subtraction relative to addition. Unified descriptions of these and other diverse implementations of redundant arithmetic can be viewed as evidence for the generality and usefulness of the WTS paradigm.

Research on the representational power of twit-based encodings and their various applications is continuing. Problems currently being addressed include developing theories for general WTS representations (including twit-based formulation of digit-set conversions, necessary, and sufficient conditions for constant-time WTS conversion, and representability of arbitrary digit sets), and deriving design details for twit-based multipliers, dividers, and other arithmetic circuits. Design of application-specific units for digital signal processing applications and cryptography is also envisaged.

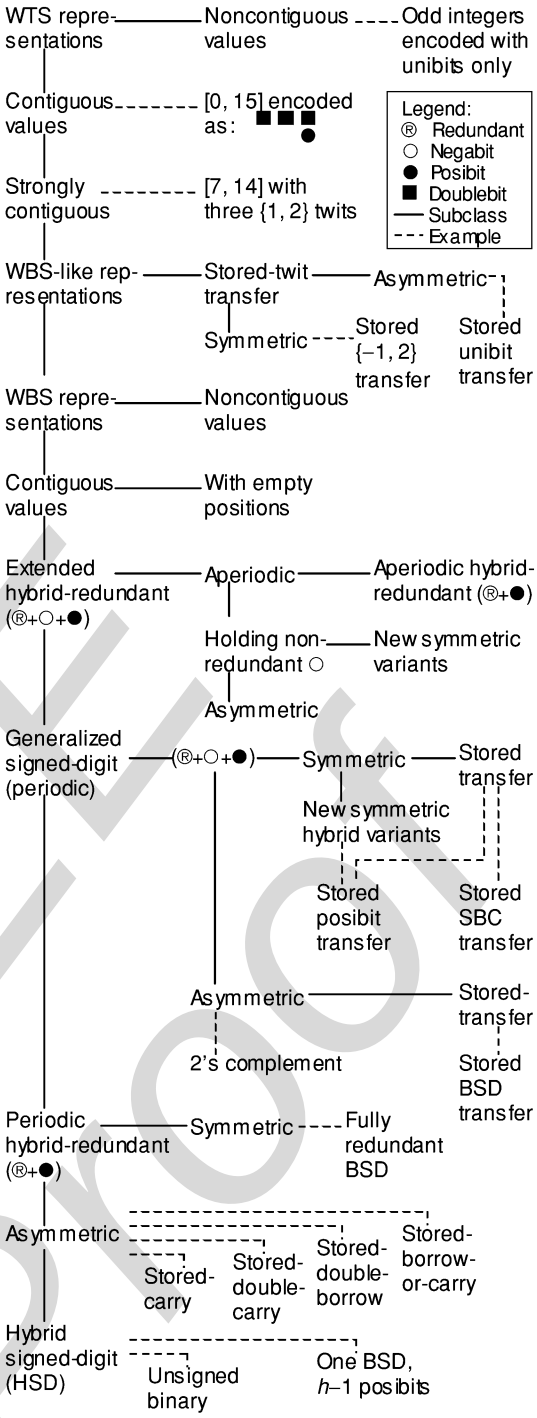


Fig. 13. Hierarchy of number representations resulting from WTS encoding (tree branches go from left to right and top to bottom).

Appendix A

Theorem 1 (Twit FA): A standard full-adder cell receiving as inputs any three equally weighted, equigap, bias-encoded twits with values in $\{\lambda_i, \lambda_i + \gamma\}, i = 1, 2, 3$, produces carry and sum twits which have the same gap as the inputs and with bias values λ_c and λ_s satisfying $2\lambda_c + \lambda_s = \lambda_1 + \lambda_2 + \lambda_3$.

Proof: We describe the operation of an equigap twit full-adder and show that it can be implemented by a standard binary full-adder. Form the sum of three equally weighted equigap bias-encoded twits $\lambda_1+\gamma x_1, \lambda_2+\gamma x_2, \lambda_3+\gamma x_3$

$$(\lambda_1+\gamma x_1)+(\lambda_2+\gamma x_2)+(\lambda_3+\gamma x_3) = (\lambda_1+\lambda_2+\lambda_3)+\gamma(x_1+x_2+x_3).$$

Let c and s be the carry and sum outputs of a standard full-adder, with the encoding bits x_1, x_2, and x_3 as inputs, and select biases λ_c and λ_s such that $2\lambda_c+\lambda_s = \lambda_1+\lambda_2+\lambda_3$. Substituting $2\lambda_c+\lambda_s$ for $\lambda_1+\lambda_2+\lambda_3$ and $2c+s$ for $x_1+x_2+x_3$ in the right-hand side of the equation above, we get

$$(2\lambda_c+\lambda_s)+\gamma(2c+s) = 2(\lambda_c+\gamma c)+(\lambda_s+\gamma s).$$

Note that selection of λ_c and λ_s is always possible; if $\lambda_1+\lambda_2+\lambda_3$ is even (odd), say equal to $2j(2j+1)$, then $\lambda_s = 2i\,(2i+1)$, and $\lambda_c = j-i$, for all integers i and j. Design peculiarities may guide the latter choices. For example with equibias input twits, the output carry and sum twits can be made to have the same bias. □

Corollary 1 (Twit Compressor): A standard binary $(\nu;\mu)$-compressor receiving ν equally weighted, equigap twits in position i produces μ twits with the same gap in positions i to $i+\mu-1$, such that inputs and outputs have the same collective values. Moreover, because any (multicolumn) posibit compressor can be implemented by a collection of standard full-adders, such a compressor may receive equigap twits, in lieu of input posibits, and produce twits with the same gap where one normally would see output posibits. □

Corollary 2 (In-Place Reduction of Twits): Three equally weighted, equigap, bias-encoded twits $\lambda_1+\gamma x_1, \lambda_2+\gamma x_2$, and $\lambda_3+\gamma x_3$, may be reduced, by a full-adder, to two equally weighted twits, one with a doubled-gap (i.e., the carry output), and one with the original gap (the sum output), such that

$$(2\lambda_c+2\gamma c)+\lambda_s+\gamma s = (\lambda_1+\lambda_2+\lambda_3)+\gamma(x_1+x_2+x_3).$$

Furthermore, the equally weighted twits $\{2\lambda_c, 2\lambda_c+2\gamma\}$, having an even bias, and $\{\lambda_s, \lambda_s+\gamma\}$ can be replaced with $\{2\lambda_c+1, 2\lambda_c+1+2\gamma\}$ and $\{\lambda_s-1, \lambda_s-1+\gamma\}$, respectively. □

Theorem 2 (Gaps in Representation): The maximum distance between consecutive integer values in an ordered collection of integers representable by a set of $m \geq 1$ equally weighted twits, given the twit gaps in descending order $\{\gamma_{m-1}, \ldots, \gamma_1, \gamma_0\}$, is

$$d^{\max} = \max\{(\gamma_j - \Gamma_{(j)}) \,|\, 0 \leq j < m\}, \quad \text{where } \Gamma_{(0)} = 0 \quad \text{and} \quad \Gamma_{(j)} = \textstyle\sum_{0\leq i<j}\gamma_i.$$

Proof (by Induction on m): Let $\{\lambda_i, \lambda_i+\gamma_i\}$ denote the values of a twit. The base case $m=1$ is obvious, given that the formula yields $d^{\max} = \max\{(\gamma_j-\Gamma_{(j)}) \,|\, 0 \leq j < 1\} = \gamma_0$. Now assume that $m-1$ twits represents the ordered set of integers $\Psi_{m-1} = \{\Lambda_{(m-1)}, \ldots, \Lambda_{(m-1)}+\Gamma_{(m-1)}\}$, per the theorem's statement, with $d^{\max} = \max\{(\gamma_j-\Gamma_{(j)}) \,|\, 0 \leq j < m-1\}$, where $\Lambda_{(m-1)} = \sum_{0\leq i<m-1}\lambda_i$ and $\Gamma_{(m-1)} = \sum_{0\leq i<m-1}\gamma_i$. We now include $\{\lambda_{m-1}, \lambda_{m-1}+\gamma_{m-1}\}$ in the set of twits. The represented values for the m-twit collection are

$$\Psi_m = \{(\lambda_{m-1}+v) \,|\, v \in \Psi_{m-1}\} \cup \{(\lambda_{m-1}+\gamma_{m-1}+v) | v \in \Psi_{m-1}\}.$$

The value of $d^{\max}$ within each of the subcollections in Ψ_m remains the same as that of Ψ_{m-1}. If the ranges of values in the two parts of Ψ_m overlap, then $d^{\max}$ for Ψ_m remains the same as that of Ψ_{m-1}, which together with $\gamma_{m-1} \leq \Gamma_{(m-1)}$ (due to the overlap), meet the condition of the theorem's statement. Otherwise the new $d^{\max}$ is the maximum of the old one, and the distance between the minimum value of the second part of Ψ_m and the maximum value of the first part, that is, $\lambda_{m-1}+\gamma_{m-1}+\Lambda_{(m-1)} - (\lambda_{m-1}+\Lambda_{(m-1)}+\Gamma_{(m-1)}) = \gamma_{m-1}-\Gamma_{(m-1)}$. □

Corollary 3 (Representational Contiguity of Twit Sets): A nonempty set of equally weighted twits represents an interval of integers (i.e., $d^{\max}=1$) iff, given the gaps in descending order $\{\gamma_{m-1}, \ldots, \gamma_1, \gamma_0\}, \gamma_0 = 1$ and $\gamma_j \leq 1+\Gamma_{(j)}$ for $0 < j \leq m-1$. □

Corollary 4: The interval of integers represented by m twits meeting the conditions of Corollary 3, and $\Lambda_{(m)} \leq 0 \leq \Lambda_{(m)}+\Gamma_{(m)}$, is representable by a collection of $-\Lambda_{(m)}$ negabits and $\Lambda_{(m)}+\Gamma_{(m)}$ posibits, where the redundancy index (Definition 4) of the interval is $\Gamma_{(m)}-1$. □

Theorem 3 (Size of Twit Representation): A contiguous interval $[\Lambda_{(m)}, \Lambda_{(m)}+\Gamma_{(m)}]$ of integers is representable by at least $m = \lceil \log_2(\Gamma_{(m)}+1) \rceil$ equally weighted twits $\{\lambda_i, \lambda_i+2^i\}, 0 \leq i \leq m-2$, and $\{\lambda_{m-1}, \lambda_{m-1}+\gamma_{m-1}\}$ with $\gamma_{m-1} = \Gamma_{(m)}+1-2^{m-1}$.

Proof: We have $2^{m-1} \leq \Gamma_{(m)} = \sum_{0\leq i<m}\gamma_i \leq 2^m-1$ by the Theorem's conditions. We choose λ_i, for $0 \leq i \leq m-1$, such that $\Lambda_{(m)} = \sum_{0\leq i\leq m-1}\lambda_i$. Such a set of m twits represents the interval $[\Lambda_{(m)}, \Lambda_{(m)}+\Gamma_{(m)}]$ for it meets the conditions of Corollary 4. We prove that m is minimal by contradiction. Suppose there is a collection of $(n < m)$ twits with gaps in descending order $\{\gamma'_{n-1}, \ldots, \gamma'_1, \gamma'_0\}$, collectively representing $[\Lambda_{(m)}, \Lambda_{(m)}+\Gamma_{(m)}]$. Then, we have: $2^{m-1} \leq \sum_{0\leq i<n}\gamma'_i = \Gamma_{(m)} \leq 2^m-1$. For the latter inequality to hold, there must be at least one twit gap satisfying $\gamma'_j > 2^j$. Assume that γ'_j is the first such twit gap (the one with the smallest index). But by Corollary 4 we have

$$\gamma'_j \leq 1+\gamma'_0+\gamma'_1+\gamma'_2+\cdots+\gamma'_{j-1} \leq 1+1+2+4+\cdots+2^{j-1} = 2^j.$$

The derived constraint $\gamma'_j \leq 2^j$ clearly contradicts the requirement $\gamma'_j > 2^j$. □

Corollary 5: A set of m equally weighted twits with gaps $\gamma_i = 2^i$, for $0 \leq i \leq m-1$, represents an interval of 2^m integers with maximal encoding efficiency $e=1$ (see Definition 4). □

Theorem 4: An interval $[\Lambda^+, \Lambda^+ + \Gamma^+]$ of integer values containing $\Gamma^+ + 1$ consecutive integers is representable by a WBS encoding with redundancy pattern $\rho_{k-1}\ldots\rho_1\rho_0$ iff for all indexes i in the range $0 < i < k$, we have $R_i \geq 0$.

Proof: The necessity part is easy to prove. If $R_i < 0$ for some i, then positions 0 to $i-1$ collectively represent fewer

than 2^i distinct values. At least one of the 2^i mod-2^i equivalence classes must be unrepresented among these values. Given that bits in positions i and higher can only represent multiples of 2^i, there must be gaps in the representation. We prove the sufficiency part by induction on k. Recall that m_{k-1} is nonzero by Definition 1. This leads to $m_0 > 0$, because either position 0 is the only position or else the assumed condition $R_i \geq 0$ guarantees $R_1 = \rho_0 = m_0 - 1 \geq 0$. The base case is $k = 1$; a one-position WBS representation with $m_0 > 0$ covers all integers from Λ_0 to $\Lambda_0 + \Gamma_0$. Suppose that the theorem holds for any WBS representation with at most $k-1$ positions. Let a k-position representation Ω be obtained by extending a $(k-g)$-position representation, where $g \geq 1$, with $m_{k-1} > 0$ and $m_j = 0$ for $k - g \leq j < k - 1$, that is, assume that the leftmost g components of redundancy pattern are $\rho_{k-1} - 1 - 1 \ldots - 1$. Then, by our assumptions, $R_{k-1} = R_{k-2} = \cdots = R_{k-g} \geq 0$. In particular, $R_{k-1} = (-1-1\ldots-1\rho_{k-g-1}\ldots\rho_1\rho_0)_{\text{two}} \geq 0$ leads to $-2^{k-1} + 2^{k-g} + R_{k-g} \geq 0$, or $R_{k-g} + 2^{k-g} \geq 2^{k-1}$. This implies that the interval represented by the rightmost $k-g$ positions of Ω contains at least 2^{k-1} consecutive values. These values combined with multiples of 2^{k-1} representable by the bit(s) in position $k-1$ yield a continuous interval of integers overall. □

Theorem 5 (Uniqueness of Redundancy Pattern for Strongly Equivalent Canonical WBS Encodings): Any WBS encoding with total redundancy index R and the redundancy pattern $\rho_{k-1}\ldots\rho_1\rho_0$ satisfying $R_i \geq 0$ for $0 < i < k$, and thus representing a continuous interval of integers by Theorem 1, is strongly equivalent to one or more canonical WBS encodings with a common redundancy pattern and the same total redundancy index R.

Proof: We describe the process for deriving a canonical encoding from a given WBS encoding. Scan the redundancy indexes ρ_i from the right until you find $\rho_j \geq 2$ for some $j < k-1$. If no such position exists, the encoding is already in the desired canonical form; we will show later that $\rho_i \geq 0$ for $0 \leq i \leq j-1$. If you find $\rho_j \geq 2$, take three of the bits in position j and make the substitution shown in Fig. 3. This does not change the set of values representable (which preserves the total redundancy index R) and reduces ρ_j by 2. Repeating this process eventually leads to $\rho_j \leq 1$ for $0 \leq j < k-1$. To show that the resulting redundancy indexes satisfy $\rho_j \geq 0, 0 \leq j < k-1$, we note that $R_j = (-1\rho_{j-2}\ldots\rho_0)_{\text{two}}$ has a value of -1 when all the redundancy indexes assume the maximal value of 1. We can prove the uniqueness of the redundancy pattern by contradiction. Suppose that another equivalent canonical encoding with a different redundancy pattern exist, and let l be the leftmost (most significant) position in which redundancy indexes differ. If R' is the total redundancy index for this second canonical encoding, $R - R'$ (that is, the difference between the sizes of intervals representable by the two encodings) will be nonzero, given that $R - R' \geq 2^l - (1\,1\ldots1)_{\text{two}} = 1$. □

Corollary 6: A given k-position WBS encoding is *redundant* iff in any of its strongly equivalent canonical forms, $\rho_j > 0$ for some $j < k$. □

Theorem 6 (Efficiency of Canonical WBS Encodings): Among all strongly equivalent WBS encodings, canonical encodings have the highest encoding efficiency.

Proof: We show, by contradiction, that the encoding cost $E = \sum_{0 \leq i < k} m_i$ is minimal for canonical encodings. If a canonical encoding does not have the lowest cost among all strongly equivalent WBS encodings, uniqueness of the redundancy pattern for canonical encodings implies that the lowest-cost strongly equivalent encoding must be noncanonical. This is impossible, however, because the process of transforming a WBS encoding to a canonical form (described in the Proof of Theorem 5) is solely composed of repeated applications of the substitutions shown in Fig. 5, and each such substitution reduces the encoding cost E by 1. □

Theorem 7 (Canonical Encoding With a Given Range): For an interval $[-N, P]$ of integers, that includes 0, and integer k in $[1, \lceil \log_2(N + P + 1) \rceil]$, a k-position canonical WBS encoding representing exactly $[-N, P]$ exists.

Proof: A trivial one-position WBS encoding with the given range has N negabits and P posibits, and $R = m_0 - 1 = N + P - 1$. A k-position canonical encoding equivalent to the above can be easily derived by the construction of Theorem 5. □

Corollary 7 (WBS Encoding for a GSD Representation): For any radix-2^h GSD number system with the digit set $[\alpha, \beta]$, there exists a periodic canonical WBS encoding with period h, where $1 \leq h \leq \lceil \log_2(\beta - \alpha + 1) \rceil$. □

APPENDIX B

List of symbols, abbreviations, and key terms.

′ ″ ‴	Used to distinguish entities with the same symbols
●, ○	Dot notation for bit or posibit, negabit (Fig. 2)
▣	Dot notation for unibit (Fig. 2)
■, □	Dot notation for doublebit, negadoublebit (Fig. 2)
α, β	Parameters of digit set $\Delta = \{\alpha, \ldots, \beta\}$
Γ	Gap parameters: $\Gamma_{(i)}$ (Def. 1); Γ_i, Γ_i^+, Γ^+ (Def. 4)
γ	Gap size for the twit $\{\lambda, \lambda + \gamma\}$ (Def. 1)
Δ	Digit set
ε	Effective gap (Def. 4)
Λ	Lower values: $\Lambda_{(i)}$ (Def. 2); Λ_i, Λ_i^+, Λ^+ (Def. 4)
λ	Lower value for the twit $\{\lambda, \lambda + \gamma\}$ (Def. 1)
μ, ν	Number of compressor outputs and inputs (Cor. 1)
ρ_i	Redundancy index of position i (Def. 4)
Ω	Specific WBS encoding
a, *b*	Both lower and upper case: arbitrary bit values
Biased	Twit encoding with λ encoded as 0 (Def. 2)
BSD	Binary signed-digit representation
C, *c*	Carry bit/twit
$d^{\max}$	Maximum gap in a representation
E, *e*	Encoding cost, encoding efficiency (Def. 4)
G	Transfer set (Def. 11)
GSD	Generalized signed-digit representation
h	Period (Def. 10) or power of 2 in $r = 2^h$
i, *j*	Arbitrary indices
k	Number of digit positions (Def. 3)
LSD	Least-significant digit (LSB is used in binary)
MSD	Most significant digit (MSB is used in binary)
Negabit	Two-valued digit in {–1, 0}
(*n*, *p*)	Digit composed of a negabit and a posibit
Posibit	Two-valued digit in {0, 1}; same as bit
R_i, *R*	$R_i = (\rho_{i-1} \ldots \rho_1\rho_0)_{\text{two}}$, $R = R_k$ (Def. 4)
S, *s*	Sum bit/twit
SBC	Stored-borrow-or-carry; digits in [–1, 2]
SDB	Stored-double-borrow; digits in [–2, 1]
SDC	Stored-double-carry; digits in [0, 3]
STC	Stored-triple-carry
SUT	Stored-unibit-transfer (Def. 11)
T, *t*	Arbitrary transfer values
Twit	Two-valued digit (Def. 1)
Unibit	Two-valued digit from the set {–1, 1}
u, *v*, *w*	Both lower and upper case: arbitrary bit values
WBS	Weighted bit-set encoding
WTS	Weighted twit-set encoding (Def. 4)
x, *y*, *z*	Both lower and upper case: arbitrary bit values

ACKNOWLEDGMENT

The authors gratefully acknowledge constructive comments by three anonymous reviewers that led to the removal of some redundancies and to a much clearer presentation.

REFERENCES

[1] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comp.*, vol. 10, pp. 389–400, Sep. 1961.

[2] 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, pp. 7–18, 2003.

[3] J. Duprat, Y. Herreros, and S. Kla, "New redundant representations of complex numbers and vectors," in *Proc. 10th IEEE Symp. Computer Arithmetic*, Jun. 1991, pp. 2–9.

[4] H. Fahmy and M. J. Flynn, "The case for a redundant format in floating point arithmetic," in *Proc. 16th IEEE Symp. Computer Arithmetic*, 2003, pp. 95–102.

[5] 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.

[6] ——, "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.

[7] G. Jaberipur and M. Ghodsi, "High radix signed digit number systems: Representation paradigms," *Scientia Iranica*, vol. 10, no. 4, pp. 383–391, Oct. 2003.

[8] G. Jaberipur, "Frameworks for the exploration and implementation of generalized carry-free redundant number systems," Ph.D. dissertation, Tehran, Iran, Dec. 2004. Dept. Comp. Eng., Sharif Univ. of Technol..

[9] G. Jaberipur, B. Parhami, and M. Ghodsi, An efficient universal addition scheme for all hybrid-redundant representations with weighted bit-set encoding, in *J. VLSI Signal Process.*, to be published.

[10] ——, Variations on the Theme of Hybrid Redundancy for Symmetry, Area Economy, and Energy Efficiency. submitted for publication.

[11] P. Kornerup, "Digit-set conversions: Generalizations and applications," *IEEE Trans. Comp.*, vol. 43, no. 5, pp. 622–629, May 1994.

[12] ——, "Necessary and sufficeint conditions for parallel, constant time conversion and addition," in *Proc. 14th IEEE Symp. Computer Arithmetic*, Apr. 1999, pp. 152–155.

[13] D. W. Matula, "Basic digit sets for radix representation," *J. ACM*, vol. 29, no. 4, pp. 1131–1143, Oct. 1982.

[14] D. W. Matula and A. M. Nielsen, "Pipelined packet-forwarding floating point: I. Foundations and a rounder," in *Proc. 13th IEEE Symp. Computer Arithmetic*, Jul. 1997, pp. 140–147.

[15] G. Metze and J. E. Robertson, "Elimination of carry propagation in digital computers," in *Proc. Int. Conf. Information Processing*, Paris, France, 1959, pp. 389–396.

[16] A. Mignotte, J. M. Muller, and O. Peyran, "Synthesis for mixed arithmetic," *Des. Autom. Embed. Syst.*, vol. 5, no. 1, pp. 29–60, Feb. 2000.

[17] A. M. Nielsen and D. W. Matula, "Pipelined packet-forwarding floating point: II. An adder," in *Proc. 13th IEEE Symp. Computer Arithmetic*, Jul. 1997, pp. 148–155.

[18] B. Parhami, "Generalized signed-digit number systems: A unifying framework for redundant number representations," *IEEE Trans. Comp.*, vol. 39, no. 1, pp. 89–98, Jan. 1990.

[19] ——, "On the implementation of arithmetic support functions for generalized signed-digit number systems," *IEEE Trans. Comp.*, vol. 42, no. 3, pp. 89–98, Mar. 1990.

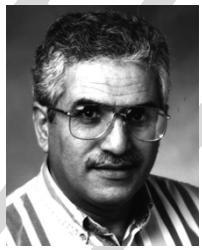
[20] D. S. Phatak and I. Koren, "Hybrid signed-digit number systems: A unified framework for redundant number representations with bounded carry propagation chains," *IEEE Trans. Comp.*, vol. 43, no. 8, pp. 880–891, Aug. 1994.

[21] ——, "Constant-time addition and simultaneous format conversion based on redundant binary representations," *IEEE Trans. Comp.*, vol. 50, no. 11, pp. 1267–1278, Nov. 2001.

[22] N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Comp.*, vol. 34, no. 9, pp. 789–796, Sep. 1985.

Ghassem Jaberipur received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, the M.S. degree in engineering (majoring in computer hardware) from the University of California, Los Angeles, the M.S. degree in computer science from University of Wisconsin, Madison, and the Ph.D. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 1974, 1976, 1979, and 2004, respectively.

Since 1979, he has been with the Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran, teaching courses in compiler construction, automata theory, design and implementation of programming languages, and computer arithmetic.



Behrooz Parhami (F'xx) received the Ph.D. degree from the University of California, Los Angeles, in 1973.

He is a 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 (1974–1988), he was 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 1979–1984. He also served as Chairman of IEEE Iran Section (1977–1986) and received the IEEE Centennial Medal in 1984.

Mohammad Ghodsi received B.S.degree in electrical engineering from Sharif University of Technology (SUT) Tehran, Iran, the M.S. degree in electrical engineering and computer science from University of California at Berkeley, and the Ph.D. degree in computer science from the Pennsylvania State University in 1975, 1978, and 1989, respectively.

He has been affiliated with SUT as a Faculty Member since 1979. Presently, he is an Associate Professor in SUT's Computer Engineering Department. His research interests include design of efficient algorithms, parallel, and systolic algorithms, and computational geometry.