2011 20th IEEE Symposium on Computer Arithmetic

A Family of High Radix Signed Digit Adders

Saeid Gorgin
School of Computer Science, Institute for Research in Fundamental Science (IPM), Tehran, Iran
Gorgin@ipm.ir

Ghassem Jaberipur
Department of Electrical and Computer Engineering, Shahid Beheshti University and School of Computer Science, Institute for Research in Fundamental Science (IPM), Tehran, Iran
Jaberipur@sbu.ac.ir

***Abstract*—Signed digit (SD) number systems allow for high performance carry-free adders. Maximally redundant SD (MRSD) alternatives provide maximal encoding efficiency among Radix-2^h SD number systems, whereby value of *h* tunes the area-time trade-off. Straightforward implementation of the conventional carry-free addition algorithm requires three O(log *h*) addition-like operations in sequence. However, there are several MRSD implementations with only one such operation. Some of them are delay optimized, but suffer from extensive hardware redundancy, while some other equally fast adders show less power/area consumption. A careful study of the latter cases hints on variety of improvement options, based on which and a new transfer computation technique, we develop a family of faster MRSD adders that consume less power/area than all the previous relevant works. They also fit efficiently within the redundant digit floating point addition scheme. However, similar to their relevant ancestor designs, suffer from an inherent property of MRSD adders; i.e., difficulty of handling hidden leading zero-digits. To remedy this problem, we use less redundant SD representations, where our transfer extraction method applies efficiently and leads to far less complex leading zero-digit detection. All the presented designs are supported by exhaustive correctness tests and performance evaluation via 0.13 micrometer CMOS technology synthesis.**

Keywords-Computer arithmetic, Carry-free addition, Maximally redundant signed-digit number system, Signed digit adder

I. INTRODUCTION

In a digital arithmetic computation it is often the case that the result of one operation, directly or via an intermediate register, serves as an operand for a subsequent operation. The computation may be implemented through the execution of a sequence of several machine instructions (e.g., compiled code for an arithmetic expression), a sequential circuit that executes an iterative algorithm (e.g., Montgomery modular multiplication [1] or digit recurrence division [2]), or a parallel computational tree (e.g., Partial product reduction tree or butterfly FFT computation [3]). Such multi-operation computations produce many partial results (e.g., $a + b$ in $a + b + c$). The same is true for single composite operations (e.g., partial remainders in digit recurrence division). In direct hardware implementations, the partial results are neither permanently stored in memory nor reported to an external device, but are saved in the register file or routed to another arithmetic hardware unit. Therefore, actually free of constraints imposed by conventional representation standards (e.g., [4]) and the word length of memory, the design engineer can decide on the most suitable number system and encoding for representation of such intermediate results. The appropriate choice, which affects the design and implementation of the arithmetic circuits, is ruled by the prescribed design goals such as speed, power and area. Redundant number systems and encodings are widely used for such purpose [5]. For example, the radix-r ($r > 2$) signed digit (SD) number systems [6] with digit set $[-\alpha, \alpha]$ provide ultra fast carry-free addition/subtraction, where carry propagation is limited within the bits of one radix-r position [7]. Faster iterative multiplication (division) is also possible via several carry-free additions (subtractions) on redundant intermediate results. Consequently, such faster basic arithmetic operations lead to faster general multi-operation computations and function evaluation.

The straightforward implementation of the carry-free addition (CFA) algorithm [7], for radix-r position i, is illustrated in Figure 1. There are three digit-parallel consecutive steps that are undertaken for all radix-r positions $0 \le i \le n-1$ of n-digit operands. The first step computes the position-sum digit $p_i = x_i + y_i$ followed in the second step by comparison of $|p_i|$ with α, which leads to deciding the value of signed carry t_{i+1}, aka transfer digit. Computation of the interim-sum digit w_i, as $-\alpha < w_i = p_i - rt_{i+1} < \alpha$, paves the way for the final step as $s_i = w_i + t_i$, where $t_i \in \{-1,0,1\}$ for more practical cases of $\alpha < r$.

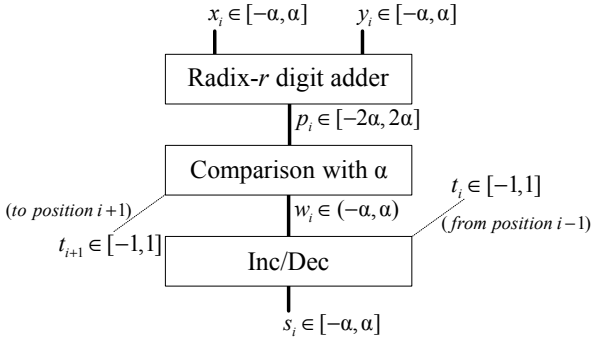


Figure 1. Illustration of the carry-free addition algorithm

The choice of $r = 2^h$ facilitates and expresses the operation of each of the latter three steps as O(log *h*) operations, where the value of h tunes the area-time trade-off. That is the higher the h, the lower the area and longer the delay for the same range of represented numbers. The case of diminished-radix SD aka maximally redundant SD (MRSD), with $\alpha = 2^h - 1$, leads to maximum encoding efficiency, where the digits in $[-2^h + 1, 2^h - 1]$ can be represented by only $h + 1$ bits. Note that larger digit sets (i.e., for $\alpha \ge 2^h$), are not generally desirable because they require more than $(h + 1)$ bits per digit (thus reducing the encoding efficiency) and also wider digit sets for representing the transfer values that complicates the adder circuitry [8].

1063-6889/11 $26.00 © 2011 IEEE
DOI 10.1109/ARITH.2011.24

Improving the performance of MRSD addition has been the subject of several research reports [9], [10], [11], [12]. In particular, the redundant digit floating point MRSD addition scheme of [10] faces the difficulty of handling hidden leading zero-digits; a problem that is not shared by the SD adders based on less redundant radix-2^h digit sets (e.g., $[-2^h+2, 2^h-2]$). However, we have not encountered any efficient addition scheme for such digit sets. On the other hand, in the case of non-power-of-two radices, decimal SD adders are practically desirable, where [13], [14], [15] and [16] have offered such decimal adders for $\alpha = 6$, 7, 9 and 7, respectively. The latter work and also [17] use a weighted bit-set partitioning technique for transfer extraction and interim-sum computation. In this paper, we aim to adapt the latter technique for radix-2^h cases, whether MRSD or less redundant.

The rest of this paper provides in Section II, a background on SD number systems, the conventional carry-free addition algorithm, and a brief description of previously reported radix-2^h MRSD adders. The proposed MRSD addition scheme with several implementation alternatives and the corresponding logical circuits are offered in Section III. Less redundant cases, which lead to better performance in leading zero-digits detection, are discussed in Section IV. All the studied designs in Sections III and IV are checked for correctness in Section V, where the synthesis results via Synopsys Design Compiler are used to choose the best MRSD adder and evaluate the performance of the proposed less redundant SD adder. Comparison with the relevant previously reported MRSD adders is taken up in Section VI and finally, concluding remarks are left for Section VII.

II. BACKGROUND

The signed digit (SD) number system, with balanced digit sets, was originally invented by Avezienis [6] and later generalized by Parhami [7] to include asymmetric digit sets. A balanced integer signed digit set $[-\alpha, \alpha]$ leads to carry-free addition if and only if $\alpha \geq \lceil (r+1)/2 \rceil$, where a generated carry in any radix-r position i does not propagate beyond position $i+1$. In more practical cases, where $r = 2^h$, the latter constraint on α leads to $\alpha \geq 2^{h-1}+1$. Therefore, cardinality $\xi = 2\alpha + 1$ of the digit set is at least equal to $2^h + 3$, which requires a minimum of $h+1$ bits for representing each digit. Keeping to the latter lower bound, for efficiency sake, would put an upper bound on the cardinality of the digit set as $\xi \leq 2^{h+1}$, which implies $\alpha \leq 2^h - 1$.

The SD addition scheme of Figure 1 can be described by the function $S(X, Y)$, where the operands and sum are k-digit radix- 2^h numbers $X = x_{k-1} \cdots x_0$, $Y = y_{k-1} \cdots y_0$ and $S = s_{k-1} \cdots s_0$, respectively, such that $-\alpha \leq x_i, y_i, s_i \leq \alpha$ $(0 \leq i \leq k-1)$ and $2^{h-1}+1 \leq \alpha \leq 2^h - 1$. Note that the constraint $-2^{h+1} + 2 \leq -2\alpha \leq p_i = x_i + y_i \leq 2\alpha \leq 2^{h+1} - 2$ leads to $-1 \leq t_{i+1} \leq 1$, by the computation of Steps I and II.

Function $S(X, Y)$

For $i = 0$ to $k-1$ perform the following in parallel:

I. $p_i = x_i + y_i$.

II. $t_{i+1} = \begin{cases} -1 & \text{if } p_i \leq -\alpha \\ 0 & \text{if } |p_i| < \alpha \\ 1 & \text{if } p_i \geq \alpha \end{cases}$,$w_i = p_i - 2^h t_{i+1}$.

III. $s_i = w_i + t_i$. ◄

In general, computation of p_i, comparison of p_i with α, and the final computation of s_i are at best O(log h) operations.

Several methods with the aim of reducing the number of consecutive O(log h) operations have been proposed for the case of $\alpha = 2^h - 1$ (i.e., MRSD). Since the working digit set $[-\alpha, \alpha]$ is a subrange of $[-2^h, 2^h - 1]$, it can be represented by the $(h+1)$-bit two's complement (2CL) encoding, where the only invalid code word represents -2^h. One drawback of the MRSD number systems is the complexity of detection of leading zero-digits that is required in the normalization stage of floating point addition. For example, the MRSD floating point addition scheme of Fahmy and Flynn [10] recognizes the latter as one of the two major challenges in their research. Furthermore, their work leans on extensive hardware redundancy to speculate the sum digits corresponding to the three different possible values of the signed carry (i.e., $t_i \in \{-1, 0, 1\}$), thus reducing the latency to the order of one O(log h) operations. We categorize other fast MRSD addition schemes, that we have encountered, based on how they compute the transfer digits: estimation and subsequent correction or accurate extraction.

- **Estimation**: Recalling Function $S(X, Y)$, we note that $-2^h + 2 \leq -2\alpha \leq p_i = w_i + 2^h t_{i+1} \leq 2\alpha \leq 2^{h+1} - 2$. The latter optimistically suggests that we estimate t_{i+1} as the signed carry-out of addition in Step I such that $t_{i+1} = \lfloor p_i / 2^h \rceil$. Unfortunately however, the latter leads to wrong interim-sum digit (i.e., $w_i = 2^h - 1$) in case of $|p_i| = 2^h - 1$. Therefore, the estimated signed carry needs to be corrected in the latter two exceptional cases. Besides, there are still two O(log h) addition operations to be performed in sequence. However, we have previously proposed a method that estimates the transfer digit t_{i+1} by inspecting only the most significant bits (MSBs) of x_i and y_i [12]. The estimated value hits the correct transfer digit except for few cases that are handled via transfer correction logic with two h-bit inputs, where the complexity of such logic can be problematic and undesirable for large h. This scheme is illustrated by Figure 2a.
- **Extraction**: The signed carry and interim-sum digit computation of Step II of function $S(X, Y)$ can be conceptually implemented directly via combinational logic as $t_{i+1} = \tau(x_i, y_i)$ and $w_i = \varphi(x_i, y_i)$, where no p_i is computed at the outset. This is not practical for large h unless τ and φ do not use all bits of $x_i$and y_i (Fig. 2b).

For example, in the pioneering efficient radix-2^h MRSD adder due to [9] the two MSBs of each of the two operands (i.e., $m = n = h - 1$ in Figure 2b) are used to extract the signed carry. The interim-sum digit w_i is represented as an (h+1)-bit carry-save two's complement number, where all the bits are the same as those of x_i and y_i, except for the MSBs that are obtained via correction of the original ones.

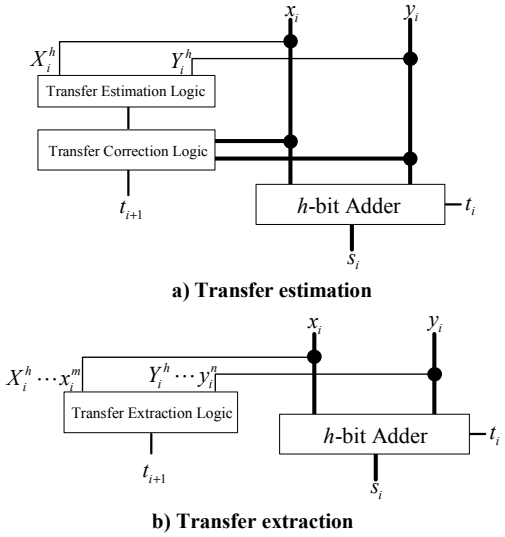


Figure 2. Abstract view of a digit-slice of radix-2^h MRSD adders

The latter work is illustrated in Figure 3, where the negatively weighted bits are distinguished by capital letters and the carries C_i^{h+1} and c_i^{h+1} are ignored. Stage II is not an operational stage and is meant to better illustrate the transfer extraction. The corresponding architecture is depicted in Figure 4, where control logic is responsible for the transfer extraction and the related corrections. One drawback of this work, which limits its application for large h, is the h-fanout due to sign extension of the signed carry. The other one is that the used extraction technique is not extendable to less redundant SD cases. However, a more general extraction technique based on sharp partitioning of the weighted bit-set [18] representing the unevaluated sum of the two operands has been used for decimal signed digit addition [16]. We found this technique equally useful for radix-2^h SD cases, whether MRSD or less redundant, in contrast to the overlapped partitioning of [9]. Therefore, in the following section, we aim to modify the latter work via focusing on, and formalizing, the sharp partitioning of the bits of the two SD operands. Furthermore, for additional performance optimization we will locate four improvement options on the Steps of Figure 3 and apply them to our designs.

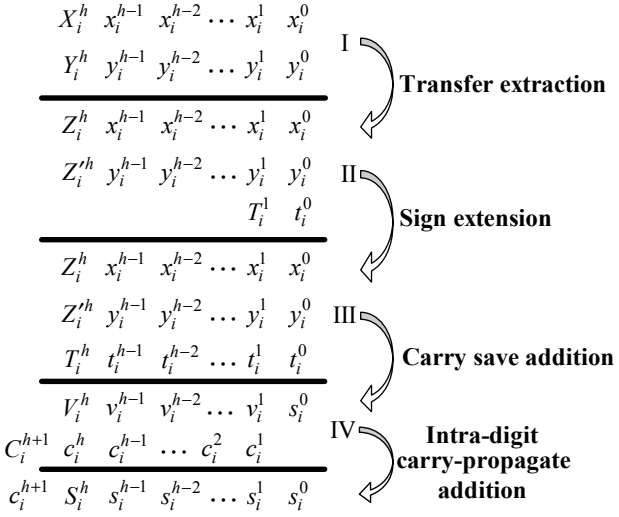


Figure 3. Illustration of MRSD carry-free addition based on [9]

III. The Proposed New MRSD Adder

A careful study of Figs. 3 and 4, hints on some potential improvement options, which are described in Section *A* under 1), 2), 3) and 4). A variety of MRSD adders with transfer extraction can be obtained via applying one or more of these design options, where all the designs are based on sharp partitioning of the bits of input operands, to be explained in Section *C*. For instance, Figure 5 depicts seven architectures that can be divided into two main groups corresponding to two encodings of the signed carry (i.e., the (n, p) and 2CL encodings).

Note that choice of $h = 4$ is without loss of generality and solely for ease of illustration. Stage III, where the black (white) dots represent positive- (negative-) weighted bits, aka posibits (negabits) [19], is not actually an operational stage.

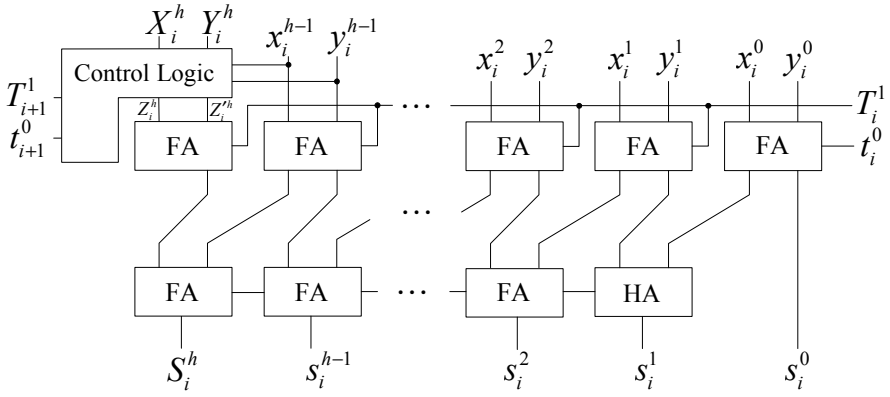


Figure 4. MRSD adder of [9]

It is meant to serve for better illustration of the fine tuning within each group, which is based on different possible groupings of bits for compression, to be further explained in Section *B*, along with the meaning of the 1-filled white dots.

A. Improvement options for MRSD addition with transfer extraction

1) Sign extension elimination: This option applies on all the designs in Figure 5. Recall Stage II of Figure 3, where the two bits x_i^1 and y_i^1 together with the negative-weighted bit T_i^1 cannot be directly handled by standard full/half adder cells. The sign extension in Stage III of Figure 3, as a straightforward solution, causes additional area and energy consumption. This could be otherwise avoided via replacing $(h-1)$ leftmost full adders (in the first row of Figure 4) with half adders. This simplification is indeed possible via use of full/half adders with inversely encoded negabits, where the logical state of a negabit with arithmetic values -1 and 0 are 0 and 1, respectively (i.e., exactly the opposite of conventional encoding) [18].

2) Earlier start of the addition process: The posibit component t_i^0 of the signed carry t_i, in Figure 3, is delivered after three logic levels. Any arrangement that may lead to earlier availability of t_i^0 allows for earlier initiation of the main carry-propagate addition. This is achieved in designs A and D of Figure 5 such that the signed carry is available as soon as the rightmost full adder receives data from above. That is, the main carry-propagate addition is not hindered by the late arrival of the signed carry.

3) (n, p) encoding of signed carry: A signed carry $t_i \in \{-1,0,1\}$, may be represented by an equally weighted negabit/posibit pair T_i^0 and t_i^0 vis-à-vis (n, p) encoding of binary signed digits [7]. This allows for two representations of 0 as $T_i^0 = -1$ and $t_i^0 = 1$ or $T_i^0 = t_i^0 = 0$. This degree of freedom can be used in adapting the logical equations in order to balance the latency of these bits (designs A and D of Figure 5) or minimizing the latency of one that is needed earlier (designs B and C of Figure 5).

4) Simplification of the control logic: The transfer extraction of Figure 3 effectively replaces the most significant bits of the two operands with four bits of t_{i+1} and z_i; actually a boost up 2-by-4 replacement. Note that the most significant posibits, while contributing in the value of t_{i+1}, remain intact. The arithmetic value of the two negabits Z_i^h and Z'^h_i collectively represent $\{-1,0\}$ as suggested in Table 1 of [9]. Therefore, they can be reduced to one negabit. This improvement, yet an increasing 2-by-3 replacement, would simplify the control logic of Figure 5 and replace one of the full adders with a half adder. However, as illustrated in stage II of Figure 5, a nonincreasing 4-by-4 replacement is possible, where the 4-bit set consisting of two most significant bits of the operands (Stage I of Figure 5) is replaced with two bits of z_i and two of t_{i+1}. The rationale to be explained in Section *C*.

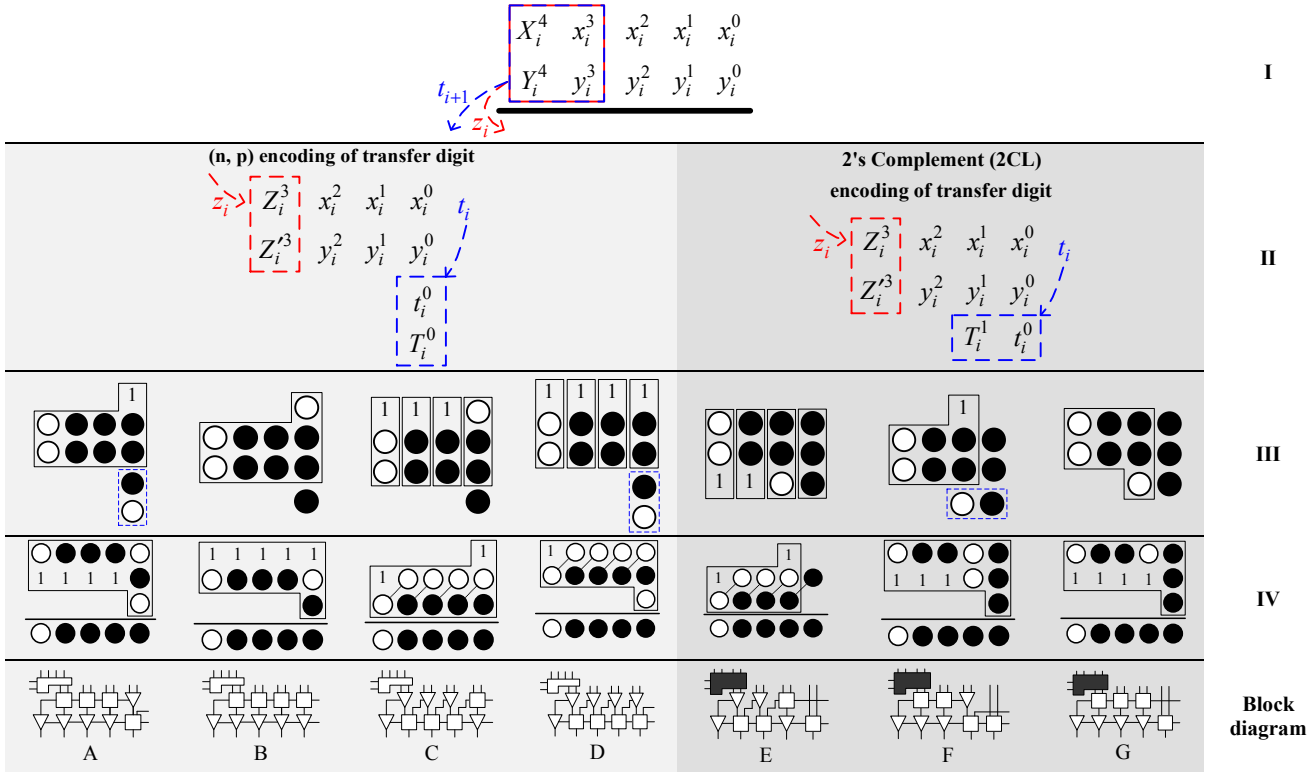


Figure 5. Seven designs of radix-16 MRSD adder based on two encodings of the signed carry

B. *Alternative MRSD adder architectures with transfer extraction*

Stage II of Figure 5 depicts two versions of transfer extraction in the MRSD addition scheme. The 2CL encoding of the signed carry (the same as in Figure 3) is depicted in the right column of Figure 5, while the one on the left uses (n, p) encoding. Stage III illustrates alternative bit-groupings for bit compression purposes, where the 3-dot rectangles are pseudo-half adders or full adders and the L-shaped dot boxes are carry propagate adders (CPA). The three designs labeled as C, D and E use carry-save adders with one input being constant 1 (i.e., enforced inversely encoded negabit with arithmetic worth of 0) where necessary.

The other four designs use intra-digit CPA. This leads to Stage IV, where all the seven designs use intra-digit CPAs to compute the 2CL sum digits. Note that, as is evident from the corresponding miniature circuit diagrams, function of the two CPAs (where applicable) overlap in time. The squares, L-shape black [white] boxes and triangles, within these circuits, represent full adders, 2CL $[(n, p)]$ transfer extraction logic and pseudo-half adders respectively. The latter is functionally equivalent to an enforced-carry full adder, which is no more complex than a standard half adder. The carry-save adders (CSA) and CPAs are configured with standard full adders and pseudo-half adders.

C. *Transfer extraction by weighted bit-set partitioning*

In this section we put forward the details of transfer extraction used in all the seven designs A-G of Figure 5. Theorem 1 proves that the signed carry can be accurately extracted from the bit partition consisting of the two most significant bits of both MRSD operands. Stage I of Figure 5 is reproduced in the left part of Figure 6, for the radix-2^h and (n, p) encoding of the signed carry t_{i+1} (2CL is similar). The collective value of the delimited bit partition u_i, namely the two 2^h-weighted negabits X_i^h and Y_i^h and the two 2^{h-1}-weighted posibits x_i^{h-1} and y_i^{h-1}, belongs to the 2^{h-1}-weighted integer range $[-4, 2]$.

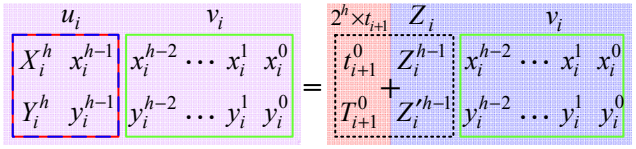


Figure 6. Decomposition of p_i ($p_i = 2^h t_{i+1} + w_i, w_i = 2^{h-1} z_i + v_i$)

These seven values can be faithfully represented by a collection of two 2^{h-1}-weighted negabits and a pair of 2^h-weighted negabit and posibit. This new bit configuration is depicted as the delimited 4-bit collection in the right part of Figure 6. Recalling $p_i = 2^h t_{i+1} + w_i$, it is also shown how the unevaluated position sum p_i can be decomposed to the signed carry t_{i+1} and a similar representation of the interim sum $w_i = 2^{h-1} z_i + v_i$. Note that the actual decomposition is applied only on bit partition u_i as $u_i = 2t_{i+1} + z_i$, such that the larger partition (for $h > 4$) v_i remains intact.

Equation sets 1 and 2, which can be derived via a simple truth table, describe the required transfer extraction logic for the just described transformation. The extracted signed carry and interim sum are valid only if $-2^h + 2 \le w_i \le 2^h - 2$. This is indeed the case as formally stated and proved below.

$$t_i^0 = \overline{X_{i-1}^h \vee Y_{i-1}^h} \vee \overline{x_{i-1}^{h-1} \vee y_{i-1}^{h-1}},$$
$$T_i^0 = X_{i-1}^h Y_{i-1}^h \quad (1)$$

$$Z_i^{h-1} = \overline{x_i^{h-1}}\left(X_i^h \vee Y_i^h\right) \vee y_i^{h-1},$$
$$Z'^{h-1}_i = \overline{y_i^{h-1}}\left(X_i^h \vee Y_i^h\right) \vee x_i^{h-1} \quad (2)$$

Theorem 1 (Validity of the extracted interim-sum): The value of interim-sum digit $w_i = 2^{h-1} z_i + v_i$, as extracted in Figure 6, belongs to the valid range $[-2^h + 2, 2^h - 2]$.

Proof: $z_i = \left\|Z_i^{h-1}\right\| + \left\|Z'^{h-1}_i\right\| \in \{-2, -1, 0\}$, where $\|x\|$ denotes the arithmetic value of x. A close examination of Equation set 2 shows that $z_i = -2$ (i.e., $Z_i^{h-1} = Z'^{h-1}_i = 0$) if and only if $X_i^h = Y_i^h = x_i^{h-1} = y_i^{h-1} = 0$. In such case w_i is valid due to the following derivations. For the other two cases (i.e., $z_i \in \{-1, 0\}$), the inequality $0 \le v_i \le 2^h - 2$ obviously leads to valid w_i values.

$$-2^h + 1 \le x_i, y_i \le -2^{h-1} - 1$$
$$\Longrightarrow -2^h - 2^h + 2 \le p_i \le -2^h - 2$$
$$\Longrightarrow -2^h + 2 \le w_i = p_i + 2^h \times (-1) \le -2 \blacktriangleleft$$

As was suggested in option a) of Section 3-*A* the signed carry is not sign extended, which indirectly allows for replacement of the bottom row full adders of Figure 4 with the chain of pseudo-half adders that are no more complex than standard half adders. Regarding option b) and also due to (n, p) encoding of signed carries, as suggested by option c), the main MRSD CPA addition in designs A and D starts immediately and no delay is imposed by the incoming signed carry.

IV. Double-Diminished SD Number System

The diminished-radix SD (i.e., MRSD) number system with digit set $[-2^h + 1, 2^h - 1]$ has been widely studied and proved to lead to highly efficient SD adders, as was discussed above. However, it suffers from the difficulty of leading zero-digits detection [20] required in redundant digit floating point addition [10]. For example, Figure 7 describes this problem for $h = 4$ (i.e., radix-16 MRSD with digit set $[-15, 15]$), where a zero-digit is actually hidden behind a leading ± 1 followed by a ∓ 15 digit. These two digits can be replaced by 0 and ± 1, respectively due to the obvious equality $\pm 1 \times 16^i \mp 15 \times 16^{i-1} = 0 \times 16^i \pm 1 \times 16^{i-1}$. The cancellation of $\pm$ 1-valued digits can continue throughout a chain of ∓ 15 digits down to a digit $l \in [\mp 14, \pm 15]$.

1 −15 −15 ⋯ −15 l ⋯ = 0 0 0 ⋯ 1 l ⋯

(a)

−1 15 15 ⋯ 15 $-l$ ⋯ = 0 0 0 ⋯ −1 $-l$ ⋯

(b)

Figure 7. Problem of hidden zero-digits for $h = 4$ [10]

It is not difficult to see that this chain of digit cancellation is an exclusive property of MRSD and does not occur in the less redundant cases of $\alpha \leq 2^h - 2$, where the SD adders are less efficient than MRSD adders [10]. However, we show in this section that the transfer extraction by sharp partitioning can lead to very efficient SD adders for $\alpha = 2^h - 2$. For ease of reference, in this paper and future works, we call such an SD number system with digit set $[-2^h + 2, 2^h - 2]$ as *double-diminished SD* (DDSD) number system.

A. *Double-diminished SD addition scheme by weighted bit-set partitioning*

The architecture of DDSD adders is generally the same as MRSD adders (Figure 5) except that the transfer extraction logic is slightly more complex. Figure 8 illustrates a 5-bit partition u_i that is necessary and sufficient for accurate transfer extraction in the case of $\alpha = 2^h - 2$. The sufficiency is proved below (Theorem 2) and the necessity follows via the insufficiency of the 4-bit partitioning, where $v_i \in [0, 2^h - 2]$. The possible case of $z_i = 0$ can lead to an invalid digit $w_i = 2^{h-1}z_i + v_i = 2^h - 2 = \alpha$.

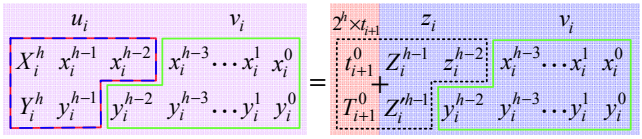


Figure 8. Decomposition of p_i ($p_i = 2^h t_{i+1} + w_i, w_i = 2^{h-2}z_i + v_i$)

Eqn. sets 3 and 4 (easily derived from a truth table) describe the required transfer extraction logic.

$$t_i^0 = \left(X_{i-1}^h \vee Y_{i-1}^h\right)\left(x_{i-1}^{h-1} \vee y_{i-1}^{h-1} \vee x_{i-1}^{h-2}\right) \vee x_{i-1}^{h-1} y_{i-1}^{h-1} x_{i-1}^{h-2},$$
$$T_i^0 = X_{i-1}^h Y_{i-1}^h \quad (3)$$

$$Z_i^{h-1} = (X_i^h \vee Y_i^h)(y_i^{h-1} \vee \overline{x_i^{h-1}}\,\overline{x_i^{h-2}}) \vee y_i^{h-1}\overline{x_i^{h-1}x_i^{h-2}},$$
$$Z'^{h-1}_i = \left(X_i^h \vee Y_i^h\right)\left(x_i^{h-1} \vee \overline{y_i^{h-1}}\,\overline{x_i^{h-2}}\right) \vee x_i^{h-1}\overline{y_i^{h-1}x_i^{h-2}},$$
$$z_i^{h-2} = x_i^{h-2} \quad (4)$$

Theorem 2 (Sufficiency of 5-bit partitioning for radix-2^h DDSD carry-free addition): The value of interim sum digit $w_i = 2^{h-2}z_i + v_i$, as extracted in Figure 8 and based on Eqn. sets 3 and 4, belongs to the valid range $[-2^h + 3, 2^h - 3]$.

Proof: The 5-bit partitioning leads to $u_i \in [-8 \times 2^{h-2}, 5 \times 2^{h-2}]$, $v_i \in [0, 3 \times 2^{h-2} - 2]$ and $z_i \in [-4,1]$. It is not difficult to see that w_i falls within the stated range for $z_i \in [-3,0]$. Regarding the boundary values −4 and 1, the following explanations are in order:

- $\boldsymbol{z_i = -4}$: Careful examination of Eqn. set 4 reveals that z_i assumes the lower bound value only when $Z_i^{h-1} = Z'^{h-1}_i = z_i^{h-2} = 0 \implies X_i^h = Y_i^h = x_i^{h-1} = y_i^{h-1} = x_i^{h-2} = 0$. The latter leads to $u_i = -2 \times 2^h$. Therefore, given that $u_i + v_i \geq 2 \times (-2^h + 2)$, we reach at $v_i \geq 4 \implies w_i \geq -2^h + 4$.
- $\boldsymbol{z_i = 1}$: Similar examination of Eqn. set 4, as in the above case, leads to Eqn. set 5.

$$x_i^{h-1} = y_i^{h-1} = x_i^{h-2} = 1, X_i^h \vee Y_i^h = 1 \quad (5)$$

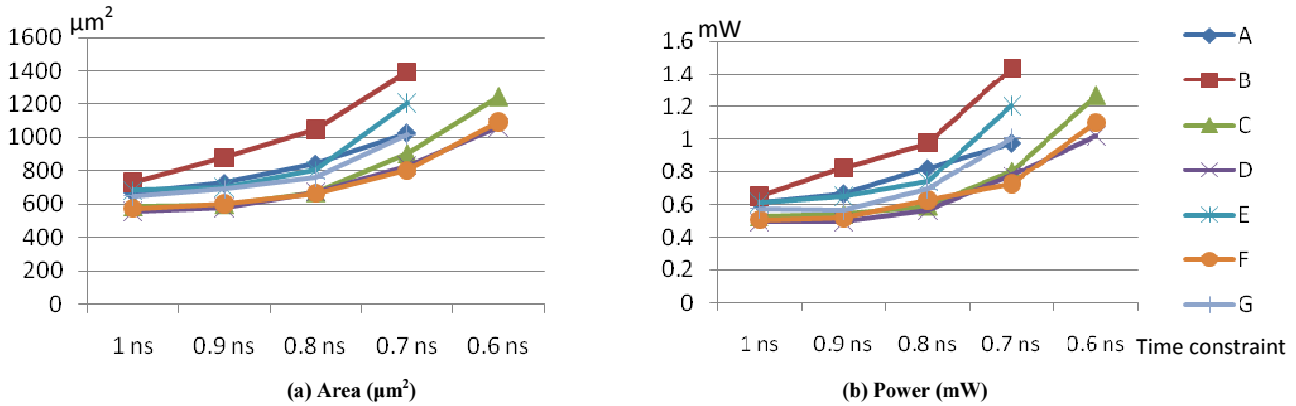
Furthermore, observe that $v_i \leq 3 \times 2^{h-2} - 2 \implies w_i = 2^{h-2}z_i + v_i = 2^{h-2} + v_i \leq 2^h - 2$. Therefore, the only invalid case is $w_i = 2^h - 2 \implies v_i = 3 \times 2^{h-2} - 2$. Fortunately, the latter cannot occur since otherwise Eqn. set 5 and the upper bound for v_i imply that all the posibits of digits x_i and y_i are 1. Given that $-2^h + 2 \leq x_i, y_i \leq 2^h - 2$, the latter leads to $X_i^h \vee Y_i^h = 0$, which violates Eqn. 5. ◄

Note that the computed w_i in cases of $z_i < 1$ does not fully cover the range of $[-2^h + 3, 2^h - 3]$ as stated in Theorem 2. While this has no harm for validity of the theorem, it is important to show that w_i can indeed assume the boundary values of the given range. This is required for preservation of the DDSD range under the proposed addition scheme. To guarantee the range preservation property, the truth table for derivation of Eqn. sets 3 and 4 is set up with a useful imprecision in the particular case of $u_i = 2^{h-2}$, where the extracted signed carry t_{i+1} can assume either values 0 or 1. We opt to let $t_{i+1} = 0$ (1) for the case of opposite (equal) sign operands. For example, consider the case of $x_i = -1 \implies (X_i^h = 0$ and $x_i^{h-1} = x_i^{h-2} = 1)$ and $y_i = 2^h - 2 \implies (Y_i^h = y_i^{h-1} = 1)$, which leads to $T_{i+1}^0 = 0$ and $t_{i+1}^0 = 1$ (i.e., $t_{i+1} = 0$), by Eqn. set 3. Therefore, $w_i = p_i = x_i + y_i = 2^h - 3$. The case of $w_i = -2^h + 3$ can also occur via possible negation operation.

V. CORRECTNESS CHECK AND SYNTHESIS

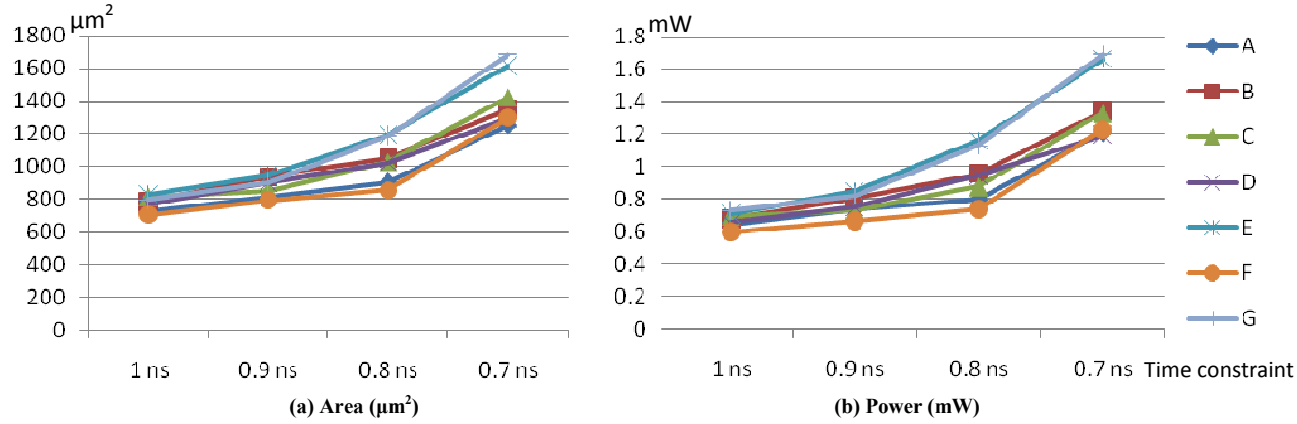
To ensure correctness of the proposed circuits in Figure 5, based on Eqn. sets 1 and 2, and the corresponding DDSD circuits based on Eqn. sets 3 and 4, we have translated all the proposed schemes to VHDL code and run exhaustive tests for $h = 4$ and 2-digit adders, where sufficiency of two digits is due to the carry-free nature of the adders and it is necessary to check the correctness of the signed carry spill over operation. These codes have also been used as input to the TSMC 0.13µm CMOS technology synthesis process via Synopsys Design Compiler. To accomplish realistic driving and loading logic, one and four inverters were used, respectively. For dynamic and leakage power measurement, Synopsys Power Compiler has been used. The results are shown in Figs. 9 and 10 for 2-digit MRSD and DDSD circuits, respectively and the A-G captions correspond to those of Figure 5. The synthesis is undertaken with decremental time constraints starting at 1.00 ns and the curves continue as far as the corresponding designs meet the time constraints. The least time constraint that is met by each design is reported in Table II.

Figures 9 and 10 and Table I suggest that, MRSD designs D and F (see Figure 9) are very competitive, while the same is evident for DDSD designs A and F (see Figure 10). However, as is more obvious in Table I, in both cases F shows marginal advantage. Therefore, in Section VI, we compare performance of the previously reported MRSD adders only with our F design.



(a) Area (µm²) **(b) Power (mW)**

Figure 9. Area (a) and power (b) of the seven proposed redundant MRSD adders



(a) Area (µm²) **(b) Power (mW)**

Figure 10. Area (a) and power (b) of the seven proposed redundant DDSD adders

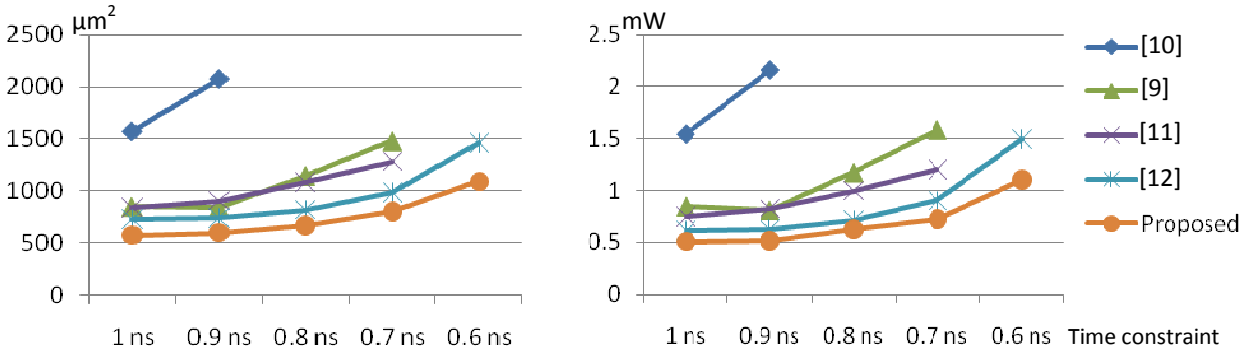
TABLE I. SYNTHESIS RESULTS FOR THE MRSD AND DDSD PROPOSED DESIGNS.

	MRSD			DDSD		
	Delay(ps)	**Power(mW)**	**Area(µm²)**	**Delay(ps)**	**Power(mW)**	**Area(µm²)**
A	627	1.87	1703	631	1.78	1688
B	640	2.19	1995	657	1.87	1805
C	585	1.58	1533	660	1.86	1844
D	568	1.76	1678	630	1.76	1745
E	678	1.80	1606	671	2.00	1975
F	**553**	**1.56**	**1469**	**606**	**1.65**	**1576**
G	645	1.36	1307	677	1.87	1745

VI. COMPARISON WITH PREVIOUS WORKS

In this section we provide the comparison results between the proposed MRSD adder (the F design) and the previously reported ones. Figure 11 depicts the area and power measures for time constraints in the range of 1.0 ns down to 0.6 ns, where none of the synthesized designs was able to meet the 0.5 ns time constraint. As it was explained in Section V the synthesis is done for radix-16 (i.e., $h = 4$) 2-digit adders, as in all the referenced works.

The proposed MRSD adder evidently shows the best area and power measures. It also meets the least time constraint among all the synthesized adders. This property is very important for the applications of redundant digit adders, where several addition operations take place before slow conversion to nonredundant representation is required. The curves of Figure 11, as in Figure 10, stop at the least time constraint that the corresponding adder can meet. Nevertheless, the exact figures for the latter and the corresponding power and area measures are compiled in Table II. The provided ratios clearly show the superiority of the proposed MRSD adder over the previous ones, except that in an absolute sense it is only slightly faster than the one in [12]. However, as is expected by careful inspection of the end parts of the corresponding curves in Figure 11, the power and area measures of the proposed adder (i.e., 1.24 mW and 1210 µm²), when synthesized with the same time constraint of 584 picoseconds, show 19.5% and 20% reduction, respectively. Furthermore, the design concept (i.e., h-dependent transfer estimation) of the adder of [12] when applied on MRSD adders with large h does not lead to considerable efficiency, while the transfer extraction cost for the proposed adders are h-independent.



(a) Area (µm²) **(b) Power (mW)**

Figure 11. Area (a) and power (b) of 2-digit MRSD adders

TABLE II. COMPARISON OF THE BEST PROPOSED MRSD ADDER WITH PREVIOUS DESIGNS.

Design Name	Delay		Power		Area	
	ps	Ratio	mW	Ratio	µm²	Ratio
[10]	862	1.56	3.23	2.08	2912	1.99
[9]	671	1.22	2.06	1.33	1889	1.29
[11]	631	1.15	2.15	1.38	1965	1.34
[12]	584	1.06	1.54	0.99	1517	1.04
Proposed	**553**	**1.00**	**1.56**	**1.00**	**1469**	**1.00**

One of the major applications of MRSD addition scheme is for the significand adder of the redundant digit floating point (FP) add/subtract unit as in [10], where the latency of significand addition is about 40% of the overall FP addition. To investigate the impact of the 56% reduced latency of the proposed MRSD adder (see Table II), we replaced the MRSD adder unit of their floating point adder architecture with our proposed one and used their Logical Effort analytical performance evaluation model to check the performance of the modified FP unit. The result is illustrated in Figure 12 for a range of significand lengths from 8 to 120 bits. The average latency reduction is about 18%. The corresponding power and area measures obtained from TSMC 0.13µm CMOS technology synthesis process via Synopsys Design Compiler, for $h = 4$ and time constraint of 4 ns is compiled in Table III, where the significand length is equivalent to that of double-precision floating point IEEE 754 standard [4].

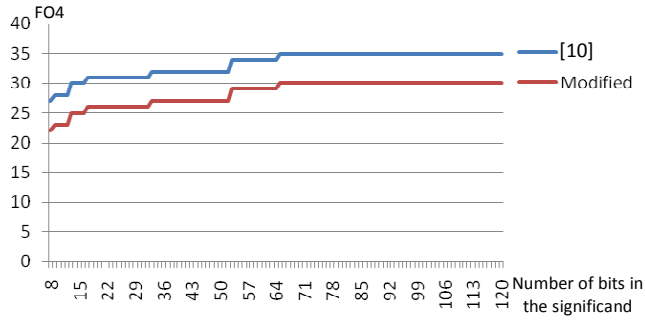


Figure 12. Comparison of the latency of the FP adder of [10] with its modified version

TABLE III. POWER AND AREA COMPARISON OF THE FP ADDER OF [10] AND ITS MODIFIED VERSION

	Power				Area	
	Dynamic (mW)	Ratio	Leakage (nW)	Ratio	(µm²)	Ratio
[10]	108.11	1.33	5.86	1.25	121291	1.26
Modified	81.09	1.00	4.69	1.00	96418	1.00

VII. CONCLUDING REMARKS

Radix-2^h maximally redundant signed digit adders are known for better performance among redundant digit adders and considerably enhance the speed of computations that include several embedded addition operations. The conventional carry-free addition algorithm for signed digit number systems has been implemented via different transfer computation methods and encodings of the signed digits and transfer digits. We categorized the previous MRSD addition schemes based on the transfer computation methods. We also recognized four improvement options, one or more of which were included in the design of seven different new MRSD adders. All the proposed designs take advantage of a special method for transfer extraction that is based on sharp weighted bit-set partitioning of the bits of the two operand digits, where the complexity of transfer logic does not depend on h. Therefore, the better synthesis results (i.e., in delay, power and area) of the proposed designs over all the previous relevant works, for $h = 4$, can lead to the conclusion of similar better results for larger values of h. In particular the proposed MRSD addition scheme when applied to the redundant digit floating point adder of [10] leads to considerable performance enhancement. Another improvement option in the latter work is to remedy the complexity of hidden leading zero-digits detection, required for handling post normalization of the significand, where the hidden nature of such digits is the exclusive property of MRSD. However, none of the transfer computation techniques presented in the previous works can be efficiently used or modified, as we elaborated on, for less redundant digit sets (e.g., double-diminished signed digit (DDSD) set $[-2^h + 2, 2^h - 2]$). Therefore, we presented an almost equally efficient addition scheme for DDSD number systems, with highest representation range after MRSD. Here again, transfer extraction complexity does not depend on h.

The impact of DDSD adders in redundant digit floating point units is an interesting subject for further research.

ACKNOWLEDGMENT

This research has been funded in part by the IPM School of Computer Science under grants CS1389-4-07, #CS1389-3-02 and in part by Shahid Beheshti University.

REFERENCES

[1] Montgomery P., "Modular Multiplication Without Trial Division," *Mathematics of Computation*, vol. 44, pp. 519–521, 1985.

[2] Ercegovac M. D., and T. Lang, *Digital Arithmetic*, Morgan Kaufman, 2004.

[3] Li Weidong, L. Wanhammar, "A pipeline FFT processor," *Proceeding of the IEEE Workshop on Signal Processing Systems (SiPS 99)*, pp. 654-662,Oct. 1999.

[4] Institute of Electrical and Electronics Engineers, *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008*, Aug. 2008.

[5] González Alejandro F., and Pinaki Mazumder, "Redundant arithmetic, algorithms and implementations," *The Integration VLSI Journal*, Vol. 30, Issue 1, pp. 13-53, Nov. 2000.

[6] Avizienis A., "Signed-digit number representations for fast parallel arithmetic," IRE *Transactions on Electronic Computers*, Vol. EC-10, pp. 389–400, Sep. 1961.

[7] Parhami B., "Generalized Signed-Digit Number System: A Unifying Framework for Redundant Number Representation," *IEEE Transactions on Computer*, Vol. 39, No. 1, pp. 89-98, 1990.

[8] Jaberipur G. and B. Parhami, "Stored-Transfer Representations with Weighted Digit-Set Encodings for Ultrahigh-Speed Arithmetic," *IET Circuits, Devices, and Systems*, Vol. 1, No. 1, pp. 102-110, Feb. 2007.

[9] Mekhallalati, M.C., and M.K., Ibrahim, "New high radix maximally-redundant signed digit adder," *Proceeding of the IEEE International Symposium on Circuits and Systems ISCAS*, Vol. 1, pp. 459–462, Jul. 1999.

[10] Fahmy H., and M.J. Flynn, "The Case for a Redundant Format in Floating-point Arithmetic," *Proceedings of the 16th IEEE Symposium on Computer Arithmetic, Santiago de Compostela, Spain*, pp. 95-102, Jun. 2003.

[11] Somayeh Timarchi, Keivan Navi, and Omid Kavehei, "Maximally Redundant High-Radix Signed-Digit Adder: New Algorithm and Implementation," *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Tampa, Florida*, pp. 97-102, May 2009.

[12] Jaberipur G, S. Gorgin, "An improved maximally redundant signed digit adder," *Computers & Electrical Engineering*, Vol. 36, No. 3, pp 491 - 502, May 2010. (doi:10.1016/j.compeleceng.2009.12.002)

[13] Svoboda, A., "Decimal Adder with Signed Digit Arithmetic," *IEEE Transactions on Computers*, Vol. C-18, No. 3, pp. 212-215, Mar. 1969.

[14] Shirazi, B., D.Y. Yun, and C.N. Zhang, "RBCD: Redundant Binary Coded Decimal Adder," *IEE Proceedings of the Computer & Digital Techniques* (CDT), Vol.36, No.2, Mar. 1989.

[15] Nikmehr H., B. J. Phillips, and C. C. Lim, "A decimal carry-free adder," *Proceeding of the SPIE Conference Smart Mater., Nano-, Micro-Smart System*, pp. 786–797, Dec. 2004.

[16] Gorgin S., and G. Jaberipur, "Fully redundant decimal arithmetic," *Proceedings of the 19th IEEE Symposium on Computer Arithmetic, Portland, USA*, pp. 145–152, Jun. 2009.

[17] Gorgin S., and G. Jaberipur, "A fully redundant decimal adder and its application in parallel decimal multipliers," *Microelectronics Journal*, Vol. 40, Issue 10, pp. 1471-1481, Oct. 2009. doi:10.1016/j.mejo.2009.07.002.

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

[19] Jaberipur G. and B. Parhami, "Posibits, Negabits, and Their Mixed Use in Efficient Realization of Arithmetic Algorithms," *Proceedings of the 15th International CSI Symposium on Computer Architecture and Digital Systems*, Iran, pp. 3–9, Sep. 2010.

[20] Schmookler Martin S. and Kevin J. Nowka, "Leading Zero Anticipation and Detection A Comparison of Methods," *Proceedings of the 15th IEEE Symposium on Computer Arithmetic, Colorado, USA*, pp. 7-12, Jun. 2001.