A High Speed Low Power Signed Digit Adder

Ghassem Jaberipur **Saeid Gorgin**
Department of Electrical and Computer Engineering, Shahid Beheshti University and
School of Computer Science, the institute of theoretical physics and mathematics(IPM), Tehran, Iran
jaberipur@sbu.ac.ir gorgin@sbu.ac.ir

Abstract: *Signed digit (SD) number systems provide the possibility of constant-time addition, where inter-digit carry propagation is eliminated. Such carry-free addition is primarily a three-step process. The special case of maximally redundant SD number systems leads to more efficient carry-free addition. This has been previously achieved based on speculation of transfer values and use of three parallel adders. We propose an alternative nonspeculative addition scheme that computes the transfer values through a fast combinational logic. The proposed carry-free addition scheme is shown to improve performance in terms of speed, power and area. The simulation and synthesis of three previous works and this work, based on 0.13 μm CMOS technology, confirms the latter claim.*

Keywords: Computer arithmetic, Carry-free addition, Signed-digit number system, Low power design, Maximal redundancy.

1. Introduction

Signed digit (SD) redundant number systems have been used in several computer arithmetic circuits [1]. SD number systems fall within the generalized signed digit (GSD) number systems that are fixed radix and contiguous number systems with digit set $[-\alpha, \beta]$, where $\alpha, \beta \geq 0$ [2]. A GSD number system is deemed redundant (nonredundant) if the redundancy index $\rho = \alpha + \beta + 1 - r$ is positive (zero), where r is the radix. The balanced SD number systems (i.e., $\alpha = \beta$) were first introduced by Avizienis [3], where $\rho = 2\alpha+1 - r > 0$. In practice r is often a power of two (e.g., 2^h), where the latter inequality leads to $\alpha \geq 2^{h-1}$. The case of $\alpha = 2^{h-1}$ corresponds to the minimally redundant digit set $[-2^{h-1}, 2^{h-1}]$, where $\rho = 1$ and at least $h+1$ bits are needed to represent a digit. But with the same number of bits α could be as much as $2^h - 1$ (i.e., nearly twice) corresponding to maximally redundant digit set $[-2^h+1, 2^h-1]$ with $\rho = 2^h - 1$. The latter is particularly attractive due to maximum range of numbers with minimum number of redundancy bits.

The main benefit of SD number systems is the possibility of constant time addition, where the latency is small and independent of the number of digits in the operands. The sum digit in position i obviously depends on the two operand digits in position i. Moreover, for the cases of $\rho = 0$ (i.e., nonredundant), $\rho = 1$, and $\rho \geq 2$, it also depends on all, two, and one less significant digit pairs, respectively. This is further explained below.

- **$\rho = 0$**: In a conventional nonredundant number system the sum digit in each position i (≥ 0) is a function of $2(i+1)$ operand digits; namely two operand digits per each of the positions $i, i-1 \ldots 1, 0$.
- **$\rho = 1$**: For minimally redundant number systems the sum digit in position i (≥ 2) is a function of the operand digits in positions i, $i-1$ and $i-2$. The constant time addition in this case is called carry-limited [2].
- **$\rho \geq 2$**: In this case, that includes the maximally redundant case of $\rho = r - 1$, the sum digit in each position i (≥ 1) depends on only four operand digits in positions i and $i-1$ [4]. The constant time addition in this case is called carry-free [2].

One drawback of redundant number systems, however, is that in practical cases of $r = 2^h$ more than h bits are needed for representation of each digit. For example, the radix-2 SD number system uses two bits to represent each digit in [–1, 1], thus doubling the storage and number of data paths. However, higher radix SD number systems (i.e., $h \geq 2$) [5] trade-off slower arithmetic for less storage and data paths [6]. The conventional carry-free SD addition algorithm (see Section 2) has three steps that each one is coarsely as slow as an h-bit adder. Therefore, paralleling or fusing these steps that could lead to faster SD addition is desirable. For example, the case of maximally redundant SD (MRSD) number systems has been attractive due to its potential for improvements leading to less latency of SD addition; noteworthy are the speculative MRSD addition in [7] and [5] and the nonspeculative approach of [8]. The speculative approach uses three parallel h+1-bit adders per each radix-2^h position.

In this paper we present an improved nonspeculative addition scheme for maximally redundant SD number systems with power of two radix $r = 2^h$. The paper is organized as follows. The conventional three-step carry-free SD addition algorithm is explained in Section 2. Some previous works on improved maximally redundant SD addition schemes, including a recent one [8], are briefly reviewed in Section 3. The contribution of this paper is presented in Section 4. The results of synthesis and simulation of this work versus three previous contributions, all based on 0.13 μm CMOS technology, are provided in Section 5. Finally we draw our conclusions in Section 6.

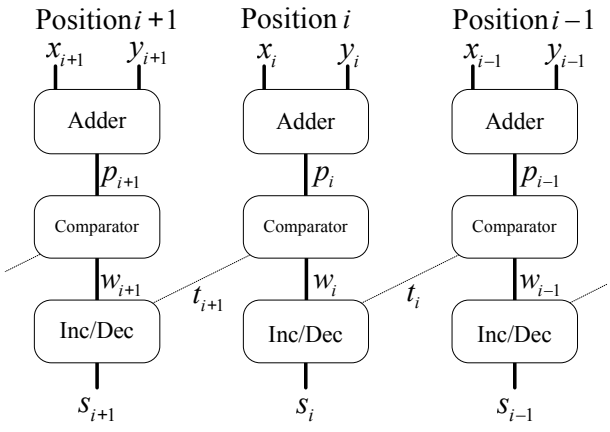

Fig. 1: The three steps of SD addition (Algorithm 1)

2. SD carry-free addition algorithm

A general carry-free addition scheme for radix-2^h SD number systems with digit set [–α, α], is described by Algorithm 1, where $\alpha > 2^{h-1}$. Fig. 1 (above) depicts a block-diagram representation of the Algorithm.

Algorithm 1 (Carry-free SD addition):

Input: Two n-digit radix-2^h SD numbers $X = x_{n-1}\ldots x_0$ and $Y = y_{n-1}\ldots y_0$, where $-\alpha \leq x_i, y_i \leq \alpha$.
Output: An (n+1)-digit radix-2^h SD number $S = s_n\ldots s_0$

I. Compute the n-digit radix-2^h SD number $P = p_{n-1}\ldots p_0 = X + Y$, by digit-parallel computation of $p_i = x_i + y_i$ for $0 \leq i \leq n-1$, where $-2\alpha \leq p_i \leq 2\alpha$.

II. Decompose p_i to transfer t_{i+1} and interim sum w_i, for $0 \leq i \leq n-1$, such that $-\alpha + 1 \leq w_i \leq \alpha - 1$, $p_i = w_i + 2^h \times t_{i+1}$, and $t_{i+1} = -1$, 0, and 1 for $p_i \leq -\alpha$, $-\alpha < p_i < \alpha$, and $p_i \geq \alpha$, respectively.

III. Form $s_i = w_i + t_i$, for $0 \leq i \leq n-1$, and set $s_n = t_n$. No new transfer will be generated in this step. ◄

Each of the Steps I and III, of Algorithm 1, involves an h-bit addition and Step II requires an h-bit comparison whose complexity is, in general, in the same order as that of h-bit addition. It would be desirable to reduce the overall latency roughly to that of two or one h-bit addition. The representation or encoding of signed digits is greatly influential on the latency of SD addition. Two's complement representation of signed digits is believed to lead to the most efficient signed digit addition schemes [5], and is the encoding of choice in all the four designs studied in this paper.

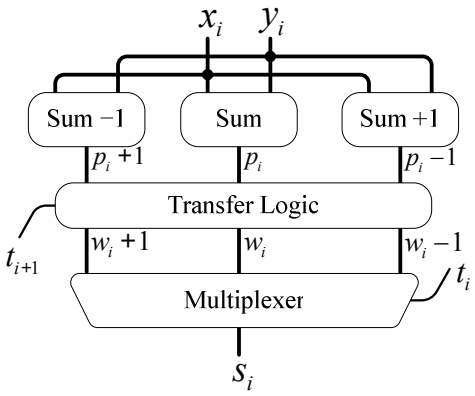

Fig. 2: The speculative SD addition

3. Efficient SD addition schemes

We briefly address three previous efficient maximally redundant SD addition schemes. Approaches a) and b), below, fuse Step I and III of Algorithm 1 in order to simultaneously compute $p_i - 1$, p_i, and $p_i + 1$ (Fig. 2, above). Then, each scheme performs Step II in its own way to compute t_{i+1}, and the three corresponding speculated sum values. However, approach c) is nonspeculative, as is depicted in Fig.3. In fact, it only computes p_i and simultaneously extracts t_{i+1} directly from x_i and y_i.

a) Fahmy and Flynn [7] have used two's complement encoding of the maximally redundant hexadecimal number system to represent redundant digit floating-point numbers. The main idea, in their SD adder is to speculatively compute $p_i - 1$, p_i, and $p_i + 1$ in parallel, decompose the sums to 16 t_{i+1} and the respected speculative s_i values (i.e., $w_i - 1$, w_i, $w_i + 1$) and let the correct s_i be selected by transfer t_i.

b) The SD addition scheme of [5] is based on an alternative treatment of Step II of Algorithm 1, where p_i is compared to 2^{h-1} instead of α.

c) The interim sum w_i and the transfer t_{i+1} may be expressed directly as functions of x_i and y_i. However, these functions are hard and inefficient to implement even for moderate values of h (e.g., eight-input functions for $h = 4$). It has been shown in [8] that t_{i+1} can generally be defined as a function of just the most significant bits of x_i and y_i, except for few cases that may be detected by a moderately simple combinational logic. This architecture is depicted in Fig. 3, where the operation of the lower adder starts as soon as the transfer t_i is available at a time that is considerably in advance of completion of operation of the upper adder.

In the next section, we follow approach c), but with some simplifications that lead to further improvements in latency, power dissipation and layout area.

4. The improved SD addition scheme

Step I of Algorithm 1 calls for the actual addition $x_i + y_i$. However, one may consider x_i and y_i as the two components of a carry-save two's complement encoding of p_i. As depicted by Fig. 4, p_i is represented by a special case of weighted bit-set (WBS) encoding [9]. In the symbolic/dot notation used in Fig. 4, posibits (i.e., normal bits) and negabits (i.e., negatively weighted bits) are represented by lowercase letters inside black dots and upper case letters inside white dots, respectively. With this encoding Step I of Algorithm 1 may actually be implemented free of cost.

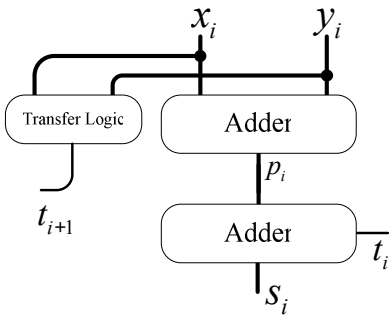

Fig. 3: The nonspeculative SD addition

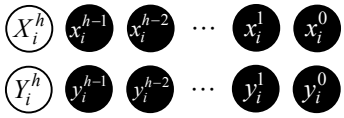

Fig. 4: Carry-save two's complement representation of the position sum p_i

The two negabits X_i^h and Y_i^h weigh 2^h, and as such may directly contribute to the value of the transfer t_{i+1}, whose weight is also 2^h. In fact, if we somehow mange to have one posibit and one negabit in position h, the bit-pair may collectively represent a valid t_{i+1} in [–1, 1]. To arrange this, observe that arithmetic value of the bit collection $\{X_i^h, x_i^{h-1}, y_i^{h-1}\}$, with respect to position h–1, falls within [–2, 2]. The same range of values may be represented by an equivalent collection of a posibit in position h and two negabits in position $h - 1$, as shown in Fig. 5. Table I shows the details of this transformation, where the target posibit and negabits are represented by primed variables and it is easily seen that $x_i^{h\prime} = \overline{X_i^h}$, $X_i^{h-1\prime} = \overline{x_i^{h-1}}$, and $Y_i^{h-1\prime} = \overline{y_i^{h-1}}$.

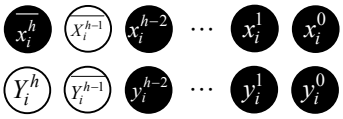

Fig. 5: Equivalent representation of position sum p_i via the transformation of Table I

Table I: Justification of transformation from Fig. 4 to Fig. 5

X_i^h	x_i^{h-1}	y_i^{h-1}	Value	$x_i^{h\prime}$	$X_i^{h-1\prime}$	$Y_i^{h-1\prime}$
0	0	0	0	1	1	1
0	0	1	1	1	1	0
0	1	0	1	1	0	1
0	1	1	2	1	0	0
1	0	0	–2	0	1	1
1	0	1	–1	0	1	0
1	1	0	–1	0	0	1
1	1	1	0	0	0	0

Let $\hat{x}_i = \overline{X_i^{h-1}} x_i^{h-2} \cdots x_i^1 x_i^0$ and $\hat{y}_i = \overline{Y_i^{h-1}} y_i^{h-2} \cdots y_i^1 y_i^0$. Then an immediate conclusion of the arrangement of Fig. 5 would be $\hat{t}_{i+1} = \overline{x_i^h} - Y_i^h$ and $\hat{w}_i = \hat{x}_i + \hat{y}_i$. Unfortunately however, it turns out that such $\hat{t}_{i+1}$ and $\hat{w}_i$ are not always correct. The exceptions are listed in Table II, and can be recognized by a flag $\varphi_i = \overline{x_i^{h-1} + \cdots + x_i^1}\ \overline{y_i^{h-1} + \cdots + y_i^1}\ \overline{x_i^0 y_i^0}$. The correct t_{i+1} and w_i are also shown in the Table.

$\hat{t}_{i+1}$ is corrected by simply subtracting 1 in all the cases that $\varphi_i = 1$ or $t_{i+1} = \hat{t}_{i+1} - \varphi_i$. This leads to the following equations that compute the constituent posibit and negabit of t_{i+1} (= $\underline{x_i^h} - \underline{Y_i^h}$).

$$\underline{x_i^h} = \overline{Y_i^h x_i^h + \varphi_i}, \qquad \underline{Y_i^h} = Y_i^h + x_i^h \tag{1}$$

Similarly, $\hat{w}_i$ gets corrected by adding 2^h. This may be effectively done by adding 1 to both $\overline{X_i^{h-1}}$ and $\overline{Y_i^{h-1}}$ in case of $\varphi_i = 1$. Note that the value of $\boldsymbol{x_i^{h-1}}$ and $\boldsymbol{y_i^{h-1}}$ before inversion and turning to negabits (see Figs. 4 and 5), in all the correction cases of Table II (i.e., $\varphi_i = 1$), is 0. This leads to $\overline{X_i^{h-1}} = \overline{Y_i^{h-1}} = -1$. Therefore, $\underline{X_i^{h-1}}$ and $\underline{Y_i^{h-1}}$ as the sign bits in the carry-save two's complement representation of w_i may be computed by Equations 2 and 3, respectively.

$$\underline{X_i^{h-1}} = \begin{cases} -1+1=0 & \text{if } \varphi_i = 1 \\ \overline{X_i^{h-1}} & \text{otherwise} \end{cases}$$

$$\Rightarrow \underline{X_i^{h-1}} = \overline{X_i^{h-1} + \varphi_i} \tag{2}$$

$$\underline{Y_i^{h-1}} = \overline{Y_i^{h-1} + \varphi_i} \tag{3}$$

The transformation from Fig. 4 to Fig. 5 and the latter corrections (i.e., Equations 1, 2 and 3) are collectively shown in the first three parts of Fig. 6. The overall delay up to this point is the delay of flag φ_i and a two-input gate in Equations 1 to 3. However, since the first $(h-1)$ bits of x_i and y_i are intact, one may start adding them at time 0 (i.e, when computation of φ_i begins) to compute the first $(h-1)$ bits of w_i. The carry out of position h–2, a posibit, and the two negabits in position h–1 feed the full adder in that position. For proper functioning of this full adder its two negabit inputs and the negabit carry-out should be inverted [10]. The result is shown in the first row of part 4 of Fig. 6. Recalling Equation 1, the two most significant bits of part 3 are extracted to form t_{i+1}. Moreover, to prepare for the last step, the transfer from position i (i.e., t_i) is converted to a two's complement number $T_i^h t_i^{h-1} \cdots t_i^0$ using the logic of Fig. 7.

Table II: The exceptions for easy extraction of transfer and the corrections

X_i^h	Y_i^h	Range of x_i and y_i	$\hat{t}_{i+1}$	$\hat{w}_i$	Exceptions for (x_i, y_i) pair	Correction	
						t_{i+1}	w_i
0	0	$x_i \geq 0, y_i \geq 0$	1	$-2^h + p_i$	(0, 0), (0, 1), and (1, 0)	0	p_i
0	1	$x_i \geq 0, y_i < 0$	0	p_i	$(0, -2^h+1)$	–1	$2^h + p_i$
1	0	$x_i < 0, y_i \geq 0$	0	p_i	$(-2^h+1, 0)$	–1	$2^h + p_i$
1	1	$x_i < 0, y_i < 0$	–1	$2^h + p_i$	None	None	None

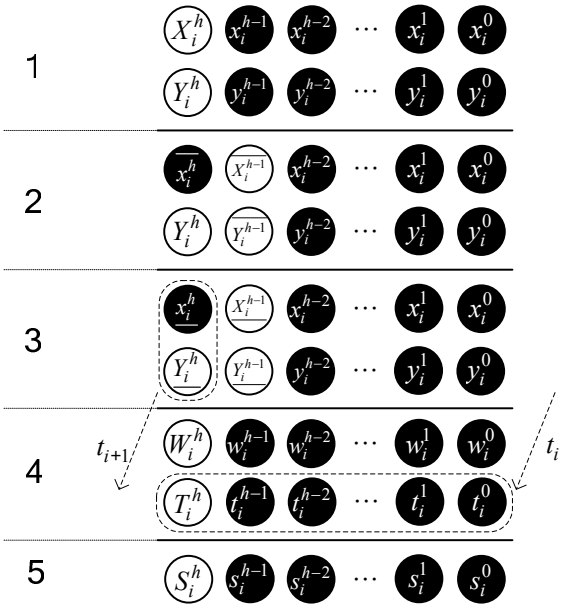

Fig. 6: Digit slice of SD addition in position i

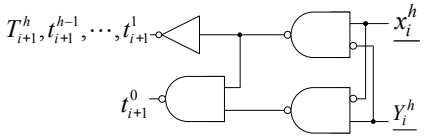

Fig. 7: Conversion of the transfer digit to an equivalent h+1-bit two's complement number

The last part of Fig. 6 is an illustration of Step III of Algorithm 1, which may be implemented using an h+1-bit two's complement adder. However, given that no new carry would be generated in this step, Equation 4 rules the most significant bit of the result, where c_i^h is the carry into position h.

$$S_i^h = \overline{c_i^h}(W_i^h + T_i^h) + W_i^h T_i^h \tag{4}$$

Fig. 8 depicts a digit slice of the overall SD adder based on Fig. 6, where the bold line is the critical delay path. However, the two full adder chains may be replaced by carry look-ahead (CLA) logic, as shown in Fig. 9, in order to reduce latency. The required CLA to replace the lower full adder chain is a simplified one, for the bits of one of the operands (i.e., $T_i^h = t_i^{h-1} = \cdots = t_i^1 = t_i$ as shown in Fig. 7) are all the same. This leads to Equation 5, where c_i^1 is the carry-into position 1 of the i^{th} digit. For large h (e.g., $h = 4k$, $k \geq 2$), a CLA tree with simplified group-generate and group-propagate signals may be used. Equation set 6 provides simplified equations for sum bits of digit slice i of the SD adder for $h = 4$, where $a = \underline{x_{i-1}^4}\,\underline{Y_{i-1}^4}$ and $b = \underline{\overline{x_{i-1}^4}}\,\underline{Y_{i-1}^4}$. A regular implementation of these equations is depicted by Fig. 10, where the lower half adder in position zero of Fig. 8 and the logic of Fig. 7 are fused for further efficiency. However, in the actual synthesis, gates with higher fan-in may be used.

$$c_i^k = t_i \sum_{j=1}^{k} w_i^j + (t_i + \prod_{j=1}^{k} w_i^j) c_i^1 \tag{5}$$

$$\begin{aligned}
s_i^0 &= w_i^0 \oplus (a + b), \\
s_i^1 &= w_i^1 \oplus \left(a\overline{w_i^0} + b w_i^0\right), \\
s_i^2 &= w_i^2 \oplus \left(a\overline{w_i^1 w_i^0} + b w_i^1 w_i^0\right), \\
s_i^3 &= w_i^3 \oplus \left(a\overline{w_i^2 w_i^1 w_i^0} + b w_i^2 w_i^1 w_i^0\right), \\
S_i^4 &= a\overline{w_i^3 w_i^2 w_i^1 w_i^0} + \overline{b w_i^3 w_i^2 w_i^1 w_i^0} W_i^4 .
\end{aligned} \tag{6}$$

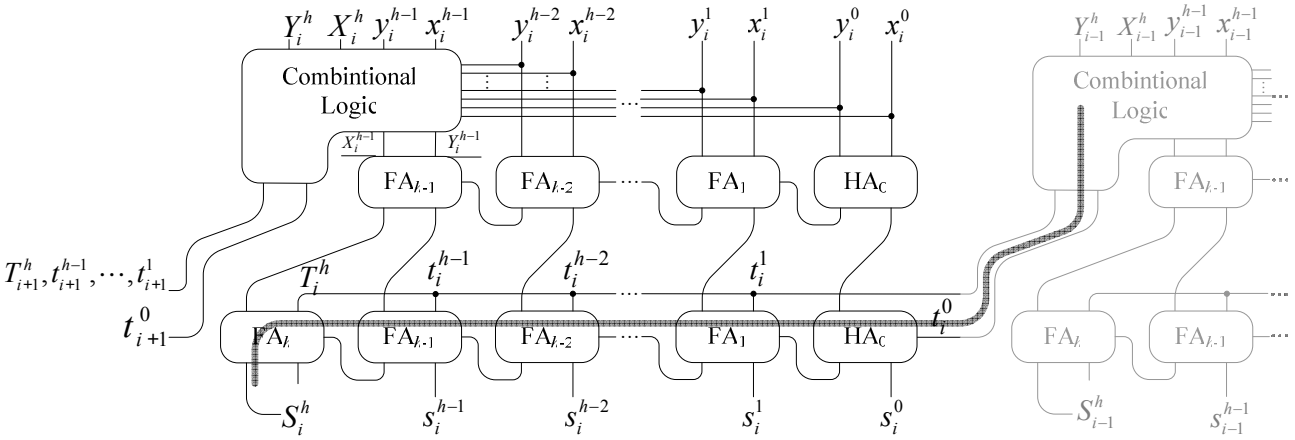

Fig. 8: Digit slice of SD adder based on Fig. 6

5. Synthesis and simulation results

SD adders operate in a digit-parallel manner. Therefore, synthesis and simulation of one digit-slice of the SD adder leads to reasonable performance measures for the whole adder. The SD adder of Fig. 9, as an improved version of Fig. 8, has been checked for correctness by exhaustive test via VHDL code of one digit-slice. To measure the performance of the adder, it was synthesized based on a 0.13 µm CMOS technology, and the results are compared in Table III together with those reported in [8]. The 34% less PDP (i.e., product of delay and power) with respect to the best previous design is quite noticeable.

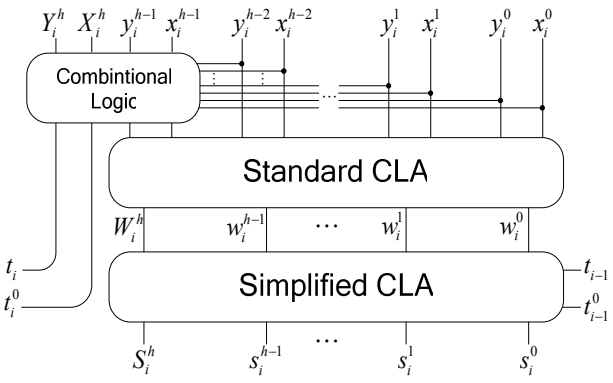

Fig. 9: The SD adder with CLA components

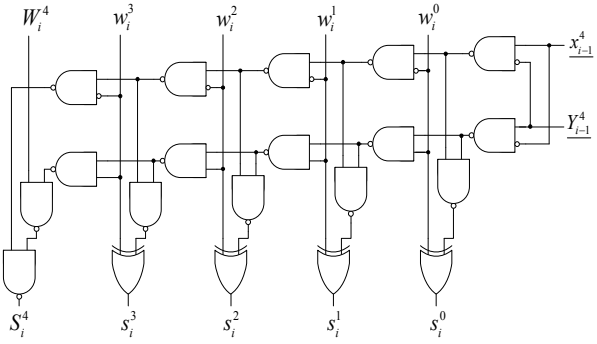

Fig. 10: The simplified CLA logic for $h = 4$ replacing the lower FA-chain of Fig. 8

6. Conclusions

We reviewed three previous efficient implementations of maximally redundant signed digit adders. Then, we proposed a new MRSD addition scheme based on carry-save two's complement encoding of positional sum-digits that are readily available by simply aligning the equally weighted digits of the operands. The first step of conventional SD addition algorithm is as such a cost-free operation. The position-sum digits are partitioned to a transfer part and an interim sum. Whereas this simple partitioning leads to invalid results, in few exceptional cases of the operands, a flag is computed to indicate the exceptions and to enforce corrections. Finally, the last step of conventional SD addition (i.e., adding the interim sum digits with the transfer coming from the next less significant digit position) is performed by a simplified carry look-ahead logic.

The new MRSD adder is checked for correctness via exhaustive tests based on VHDL code describing the adder. Synthesis and simulation of the proposed adder, based on a 0.13 µm CMOS technology, shows better performance in terms of delay, power dissipation and layout area in comparison with three previous contributions. Fig. 11, based on the results tabulated in Table III, depicts these advantages.

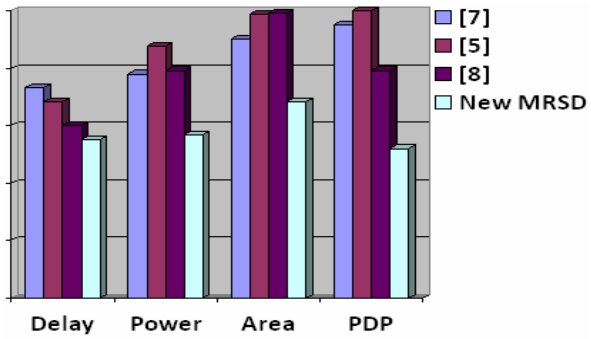

Fig. 11: Performance comparison between the new MRSD adder and three previous ones

Research is on going for further performance improvement in SD adders, and use of them in more sophisticated hardware units such as multiplication, division, and floating-point arithmetic circuits.

Table III: Simulation results for single digit MRSD adders with $h = 4$ based on 0.13 µm COMS

Design Name	Delay (ns)	Power (mW)	Area(µm^2)	Delay × power
[7]	0.61	1.95	2255.7	1.19
[5]	0.57	2.19	2473.8	1.25
[8]	0.50	1.98	2480.5	0.99
New MRSD Adder	0.46	1.42	1707.9	0.65

References

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

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

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

[4] 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.

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

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

[7] Fahmy H., and M.J. Flynn, "The Case for a Redundant Format in Floating-point Arithmetic," *Proc. 16th IEEE Symp. Computer Arithmetic*, pp. 95-102, 2003.

[8] Jaberipur G., and S. Gorgin, "A Nonspeculative Maximally Redundant Signed Digit Adder," Submitted for publication.

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

[10] Aoki T., Y. Sawada, and T. Higuchi, "Signed-Weight Arithmetic and Its Application to a Field-Programmable Digital Filter Architecture," *IEICE Trans. Electronics*, Vol. E82-C, No. 9, pp. 1687-1698, Sep. 1999.