Nonspeculative Decimal Signed Digit Adder

Liu Han, Dongdong Chen, Seok-Bum Ko, Khan A. Wahid
Department of Electrical and Computer Engineering
University of Saskatchewan, Canada
Email: {liu.han, dongdong.chen, seokbum.ko, khan.wahid}@usask.ca

***Abstract*—Decimal floating point (DFP) arithmetic has been paid more attention in recent years, since it is superior to the binary counterpart in the financial and commercial computing including currency conversion, billing system, banking and tex calculation. Many DFP arithmetic units, such as addition, multiplication, division and fused-multiply-add are not possible to achieve the high performance without a fast decimal fixed point adder. In this paper, the conventional four steps carry free signed digit addition algorithm is discussed. Furthermore, to improve the speed, we proposed a new method for the decimal SD addition and subtraction in digit set $[-9, 9]$. To evaluate the design, a VHDL model is provided and synthesized in STM 90 nm technology. The result shows that our design has a better performance on timing delay and area compared with previous designs in the same digit set.**

I. Introduction

Recently, the decimal floating point arithmetic has become more attractive in the financial and commercial computing area which includes currency conversion, billing system, banking and tex calculation, since its binary counterpart has an innate defect in aforementioned applications. The reason is that most of the decimal floating point numbers cannot be exactly represented by the binary weighted series in a finite precision, and the error can be accumulated after calculations. A detailed comparison between the decimal floating point and binary floating point arithmetic is given in [1]. Due to the importance of decimal arithmetic in financial and commercial applications, a specification was added into the IEEE standard for floating point arithmetic in 2008 [2].

Addition is the basic but the most important function among the decimal arithmetic operations. In sequential multiplication and digit recurrence division, the partial products for every iteration are accumulated by the adders. Moreover, in parallel multiplication and functional division, the partial products are reduced by the adders arranged in wallace tree structure. Since an improvement in addition can benefit to many other decimal operations, many methods and algorithms were applied to boost the performance of the decimal adder.

In signed digit (SD) number systems, there is a chance to eliminate the carry chain which causes the delay proportional to the digit width of the input. This is well known as the carry free addition. After the first published decimal signed digit adder designed by A. Svoboda in 1969 [3], some papers were presented in the last decade. H. Nikmehr et al. in [4] provided a decimal signed digit (DSD) adder in digit set $[-9, 9]$. In [5] and [6], A. Kaivani and S. Gorigin also offered good contributions on fully redundant decimal addition based on stored unibit transfer (SUT) encoding and decimal septa signed digit (DSSD) method respectively.

In this paper, the traditional decimal carry free algorithm and the possible improvement on two points are discussed in section II. A new decimal signed digit adder which performs addition and subtraction in digit set $[-9, 9]$ is proposed in section III. In section IV, based on our synthesized result in STM 90 nm library, a comparison with the previous designs is given. Finally, section V shows the concluding remarks.

II. Conventional Carry Free SD Addition

Traditionally, a decimal digit z_i, where $z_i \in \{0, 1, \ldots, 8, 9\}$, is represented in the 4-bit binary coded decimal (BCD) encoding. Alternatively, the signed digit set $\{-\alpha, -(\alpha - 1), \ldots, \alpha - 1, \alpha\}$ is applied to represent the decimal digits, and if $2\alpha + 1 > r$, where r is the radix of the number system, the number system is redundant.

The conventional SD carry free addition/subtraction algorithm is,

Algorithm I:
- Compute $P_i = X_i$ op Y_i
- Compute T_{i+1} based on the range of P_i
- Compute $W_i = P_i - r \times T_{i+1}$
- Compute $S_i = W_i + T_i$

Where P_i is the position sum, W_i is the temporary sum, T_i is the transfer digit, S_i is the result digit, r is the radix and op is the add or sub operation.

Since the transfer digit T_{i+1} to the next stage is independent on T_i from last stage, the carry chain is eliminated, and the delay is no longer related to the digit width of the input. However, this algorithm still could be improved in following two aspects:

1) Deciding T_{i+1} in parallel with P_i: Once P_i is obtained, the transfer digit T_{i+1} is generated based on the range of the position sum. Hence, an improvident delay on the critical path causes the inadequate performance. In this paper we decide the transfer digit directly on the range of the operands, X_i and Y_i. The method for range division is discussed as well in section III.

2) Calculating S_i without W_i: Since the $T_{i+1} \in \{-1, 0, 1\}$ is extracted out from P_i, a compensation in decimal addition (i.e. ± 10) is applied to correctly calculate the temporary sum, W_i. Further, the transfer digit T_i from last stage is added to

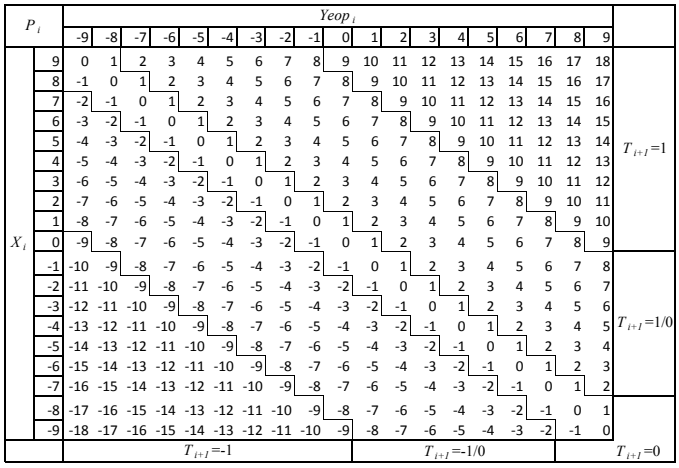

Fig. 1. The overlap region of value of T_{i+1} based on X_i and $Yeop_i$

obtain the final result S_i. In this process, these two contiguous additions imply a complicated and slower design. We proposed a way to merge them into one add operation, which results a better performance on area and delay.

III. Proposed Carry Free SD Addition

In the conventional carry free addition algorithm, to obtain the transfer digit T_{i+1}, the operands X_i and Y_i have to be added together. In this process, a carry chain limits the performance of the carry free adder. To improve it, a speculative method could be used (i.e. [4], [7] and [8]). In this method, all possible results which depend on different transfer digits T_i and T_{i+1} are calculated simultaneously and the correct one is selected by the value of the transfer digits. The redundancy on hardware in aforementioned designs implies a bigger area and higher power consumption. On the other hand, the nonspeculative method for binary signed digit addition (i.e. [9]) offered a faster and simpler design. In this section, we proposed a new nonspeculative decimal signed digit adder in digit set $[-9, 9]$ within 5 bits 2's complement encoding, which has a simple exception handling. Moreover, the result digit S_i is calculated directly based on P_i, T_i and T_{i+1} without the hardware redundancy.

A. The Algorithm

For the decimal SD addition, in Algorithm I, T_{i+1} is generated by comparing P_i with radix $\pm(\alpha - 1)$ as shown in equation (1).

$$T_{i+1} = \begin{cases} 1 & \text{if } P_i > \alpha - 1 \\ -1 & \text{if } P_i < -(\alpha - 1) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$Yeop_i = \begin{cases} Y_i & \text{if operation is } add \\ -Y_i & \text{if operation is } sub \end{cases} \quad (2)$$

To reduce the timing delay and parallelize the transfer digit generation with the position sum calculation, the temporary sum W_i and transfer digit T_{i+1} also could be directly expressed in terms of X_i and $Yeop_i$, where $Yeop_i$ is the effective Y_i shown in equation (2). In Fig. 1, the overlap region of the value of T_{i+1} implies there could be more than one way to divide the range for generating the transfer digit to left stage. One can take the advantage of this flexibility to minimize the cost of the exception detection (i.e. $P_i = \pm 9$ has to be converted into digit set of W_i, which is $[-8, 8]$) in decimal SD addition.

TABLE I
Range division based on X_i and $Yeop_i$

	Range of X_i and $Yeop_i$	T_i	W_i	Exception on X_i and $Yeop_i$
$case_1$	$X_i \geq 1, Yeop_i \geq 1$	1	$P_i - 10$	-
$case_2$	$X_i \geq 0, Yeop_i \leq 0$ $X_i \leq 0, Yeop_i \geq 0$	0	P_i	$(9,0),(0,-9)$ $(0,9),(-9,0)$
$case_3$	$X_i \leq -1, Yeop_i \leq -1$	-1	$P_i + 10$	-

In decimal SD addition, the exception handling circuit could cost an extra power consumption and even timing delay. To speed up the transfer digit generation and simplify the exception detection, a new range division method which only has four exceptions is given in Table. I. In this method, the circuit for exception detection also could be reused in range division, therefore, only the most significant bits of X_i and $Yeop_i$ and simple logic gates are needed to determine the transfer digit besides the exception detection logic.

In Algorithm I, once the transfer digit is obtained, W_i is generated by adding a decimal complement to P_i, and then, the result S_i is calculated by adding W_i with T_i. Since in this process, those two serial computations cause a limit on speed. To combine these two additions into one computation efficiently (i.e. the three operands addition $S_i = P_i - r \times T_{i+1} + Ti$), an analysis for the effect of the input range and incoming transfer bits on the decimal complement is given in Table II.

TABLE II
The decimal complement decision

	$T_i = -1$ $T_i^1 = 1\ T_i^0 = \mathbf{1}$	$T_i = 0/1$ $T_i^1 = 0\ T_i^0 = \mathbf{0/1}$	Complement $compl^{4\ldots1} =$
$case_1$	$P_i + (-11) =$ $P_i + 1010\mathbf{1}$	$P_i + (-10)/(-9) =$ $P_i + 1011\mathbf{0/1}$	1010/1011
$case_2$	$P_i + (-1) =$ $P_i + 1111\mathbf{1}$	$P_i + 0/1 =$ $P_i + 0000\mathbf{0/1}$	1111/0000
$case_3$	$P_i + 9 =$ $P_i + 0100\mathbf{1}$	$P_i + 10/11 =$ $P_i + 0101\mathbf{0/1}$	0100/0101

To generate the new decimal complement, the transfer digit T_i from least significant digit is used. Therefore, the decimal complement can be decided directly by T_i^1 and the range of input operands X_i and $Yeop_i$, then the further computation for adding T_i is removed as shown in Fig. 2 . The bold numbers in Table II show that the least significant bit of the decimal complement is equal to T_i^0. In the new decimal SD addition algorithm, since only two computations are serially connected in total, an improvement on delay could be achieved.

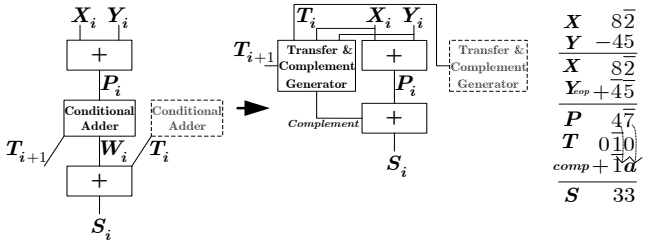

Fig. 2. Framework of the proposed two-stage decimal SD addition algorithm

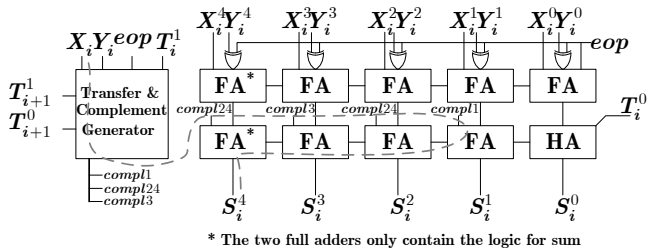

Fig. 3. The proposed nonspeculative decimal signed digit adder

B. The Hardware Implementation

To reuse the well optimized circuits in binary world as much as possible, the operands X_i and Y_i are encoded in 2's complement. Therefore, in Fig. 2, the two adding units could be implemented by the full adders connected as a ripple carry chain. To further improve the speed, the adder with high performance carry chain scheme, such as the lookahead adder, could be applied.

In this design, the exception logic is minimized to four pairs of operands detection, and to improve the speed, the operands X_i and Y_i are directly used for the exception handling. In equation (3), the signals E_p and E_n are for positive exception (i.e. $(0, 9)$ or $(9, 0)$) and negative exception (i.e. $(0, -9)$ or $(-9, 0)$) respectively.

$$\begin{aligned} E_p = &\ \Big(X_i = 0 \wedge \big((Y_i = 9 \wedge eop) \vee (Y_i = -9 \wedge \overline{eop})\big)\Big) \vee \\ &(X_i = 9 \wedge Y_i = 0) \\ E_n = &\ \Big(X_i = 0 \wedge \big((Y_i = -9 \wedge eop) \vee (Y_i = 9 \wedge \overline{eop})\big)\Big) \vee \\ &(X_i = -9 \wedge Y_i = 0) \end{aligned} \tag{3}$$

In Table I, the operands' range division for generating T_{i+1} is not right on zero, thus, the zero input should be excluded for some cases. The zero detection logic in equation (3) could be reused, and the range division logic is given as follows:

$$\begin{aligned} case_1 &= \big((\overline{X_i^4} \wedge \overline{X_i = 0}) \wedge (\overline{Yeop_i^4} \wedge \overline{Y_i = 0})\big) \vee E_p \\ case_3 &= \big(X_i^4 \wedge (Yeop_i^4 \wedge \overline{Y_i = 0})\big) \vee E_n \\ case_2 &= \overline{case_1 \vee case_3} \end{aligned} \tag{4}$$

The transfer digit only depends on the range division, and it can be obtained at the same time as P_i is ready. Thus, the critical path only pass one of the two units for transfer digit generation and position sum addition. Equation (5) shows the logics to generate the transfer bits.

$$\begin{aligned} T_{i+1}^1 &= case_3 \\ T_{i+1}^0 &= case_1 \vee case_3 \end{aligned} \tag{5}$$

According to the analysis in Table II, the decimal complements are decided by operands range and incoming transfer bits. The conditional adder with multiplexor which is controlled by T_i and $case_i$ could be applied. Nevertheless, to reduce the area, we use combinational logic shown in equation (6) to directly generate the decimal complement and connect it to the second level of binary full adders.

$$\begin{aligned} compl^4 &= case_1 \vee (T_i^1 \wedge case_2) \\ compl^3 &= case_3 \vee (T_i^1 \wedge case_2) \\ compl^2 &= compl^4 \\ compl^1 &= \overline{T_i^1 \oplus case_2} \\ compl^0 &= T_i^0 \end{aligned} \tag{6}$$

Finally, the hardware implementation of the proposed decimal SD adder is given in Fig. 3. The bold dash line is the critical path which passes through the transfer and complement logic and 4 full adders. The full adders with the asterisk only contain the logics for sum. Furthermore, in the second level of full adders, the critical path only pass through one XOR gate in the left most full adder.

C. Conversion from/to the BCD encoding

Since the decimal data stored in memory are generally encoded in BCD format, to use the decimal carry free adder, the operands in BCD encoding should be converted into the internal encoding scheme used in the proposed design. Similarly the result coming from the design needs to be converted back to BCD format before sending to the memory.

In our design, since the digit set is encoded in 5-bit 2's complement, the front conversion does not cost any logic with only one zero-bit extension. For converting from the internal 5-bit format to BCD encoding, a negative carry propagation which passes through the entire word-width logics is involved as all other redundant designs. To speed up the inverse conversion, many techniques, such as carry lookahead and prefix network, could be used. In Algorithm II, we described a conversion process based on the prefix network for high performance application.

Algorithm II [Assume result $S \geq 0$]:

• Compute generate bit (G_i) and pass bit (P_i) for each digit of the result.

$$\begin{aligned} G_i &= \begin{cases} 1 & \text{if } S_i < 0 \\ 0 & \text{otherwise} \end{cases} \quad P_i = \begin{cases} 1 & \text{if } S_i = 0 \\ 0 & \text{otherwise} \end{cases} \\ G_{i:j} &= \begin{cases} G_i & \text{if } i = j \\ G_i \vee (P_i \wedge G_{i-1:j}) & \text{if } i > j \end{cases} \\ P_{i:j} &= \begin{cases} P_i & \text{if } i = j \\ P_i \vee P_{i-1:j} & \text{if } i > j \end{cases} \end{aligned} \tag{7}$$

• Compute the negative carry C_i.

$$C_i = G_{i-1:j} \vee (P_{i-1:j} \wedge C_j) \tag{8}$$

TABLE III
THE SYNTHESIZED RESULT AND COMPARISON WITH PREVIOUS DESIGNS

	Digit set	Delay (FO4)	Ratio	Area (NAND2)	Ratio	ADP	Ratio
[4]	$[-9, 9]$	25	2.59	320	1.33	8000	3.45
[10]	$[-9, 9]$	30	3.1	290	1.21	8700	3.75
[5]	$[-8, 9]$	17.5	1.81	150	0.63	2625	1.13
Proposed	$[-9, 9]$	9.67	1	240	1	2320	1

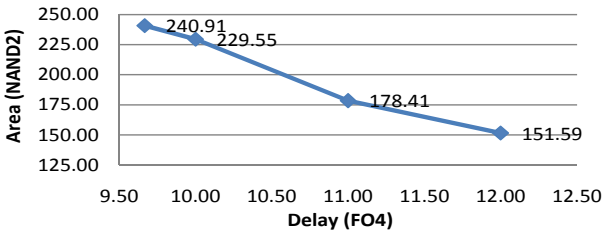

Fig. 4. Synthesis results of the proposed design

- Generate the result R_i in BCD format (Let $C_{-1} = 0$).

$$R_i = \begin{cases} S_i & \text{if } C_i = 0 \text{ and } C_{i-1} = 0 \\ S_i - 1 & \text{if } C_i = 0 \text{ and } C_{i-1} = 1 \\ S_i + 10 & \text{if } C_i = 1 \text{ and } C_{i-1} = 0 \\ S_i + 9 & \text{if } C_i = 1 \text{ and } C_{i-1} = 1 \end{cases} \quad (9)$$

In equation (9), the multiple complements could be implemented in conditional adder structure, and the finial result could be selected by the carry signals. To avoid the unnecessary area consumption, the logic also could be implemented by a decimal adder with a complement generation logic.

IV. Results and Discussion

A model of the proposed nonspeculative decimal SD adder is implemented in VHDL. We also run exhaustive tests, which include 19 X_is, 19 Y_is, 3 T_is and 2 eops, to ensure the correctness. Subsequently, the proposed design was synthesized in STM 90 nm CMOS technology with normal case parameters (i.e. 1.2v and 25°C) by Synopsys Design Compiler. The delay and area are represented in terms of fan-out of 4 (FO4) inverter's delay and a 2-input nand (NAND2) gate's area. The FO4 delay (30 ps) is measured by connecting an inverter(X1) with four inverters with same size at the output port. The NAND2(X1) area (4.4 μm^2) is directly read from the library. The area delay product (ADP) is given by FO4 delay times NAND2 area, which illustrates an intuitionistic comparison on performance.

Since our design works on the digit set $[-9, 9]$, and the operands are encoded in 2's complement, no extra front converter which converts the BCD inputs to proposed digit set is needed at all. To convert the proposed digit set to conventional BCD encoding, a carry network is needed, which is the same as other designs. In [5], the authors use the digit set $[-8, 9]$, so the front converter mentioned in their paper is included to provide a fair comparison. In Table III, our design outperforms than previous designs on the timing delay as well as on the area of the designs in paper [4] and [10]. In Fig. 4, we provided the synthesis results of our design with timing delay from 12 FO4 to 9.67 FO4. The results show that the proposed design has a better performance on delay compared with the work in [5] with the same area of 150 NAND2. Consequently, our design provides a better performance in terms of the ADP compared with all the works listed in Table III.

V. Conclusion

In this paper, we proposed a new nonspeculative decimal SD adder which calculates the operands in digit set $[-9, 9]$ with 2's complement encoding. This design determines the transfer digit directly on the input operands instead of the position sum P_i in conventional carry-free SD addition algorithm. We also analyzed the digit range of the operands to minimize the cost of exception handling. Furthermore, to improve the speed and reduce the area of the adder, we proposed a new algorithm which calculates the result digit S_i without the temporary result W_i. Finally, only two steps are in the critical path of our design instead of four in conventional carry-free algorithm.

The synthesized result in Table III shows that the proposed design provides more than 81% delay improvement over previous designs in the same digit set. In terms of the product of delay and area, this work still has more than 13% improvement than the referenced designs.

The future work is ongoing on the possibility of using this design in the decimal floating point (DFP) arithmetic, especially the DFP ADD and DFP FMA which are sensitive on the performance of the adder logic.

VI. Acknowledgement

The authors would like to acknowledge the Natural Science and Engineering Research Council (NSERC) of Canada for its support to this research.

References

[1] Cowloshaw, M. F., "Decimal Floating-Point Algorism for Computers," *in Proceedings of the 16th IEEE Symposium on Computer Arithmetic,* pp. 104-111, June 2003.

[2] "IEEE Standard for Floating-Point Arithmetic," *IEEE Working Group of the Microprocesser Standards Subcommittee,* IEEE, 2008.

[3] A. Svoboda, "Decimal adder with signed digit arithmetic," *IEEE Transactions on Computer, C-18(3),* pp. 212-215, 1969.

[4] H. Nikmehr, B. Phillips and C.C. Lim, "A decimal carry-free adder," *in Proceedings of SPIE conference on Smart Materials, Nano-, Micro-Smart Systems,* pp. 786-797, 2004.

[5] Amir Kaivani and Ghassem Jaberipur, "Fully redundant decimal addition and subtraction using stored-unibit encoding," *Integration, the VLSI journal,* pp. 34-41, 2010.

[6] Saeid Gorgin and Ghassem Jaberipur, "Fully Redundant Decimal Arithmetic," *in Proceedings of the 19th IEEE Symposium on Computer Arithmetic,* pp. 145-152, June 2009.

[7] H. Fahmy and M.J. Flynn, "The case for a redundant format in floating-point aritmetic," *in Proceedings of the 16th IEEE Symposium on Computer Arithmetic,* pp. 95-102, June 2003.

[8] Ghassem Jaberipur and Ghodsi, M., "High Radix Signed Digit Number Systems: Representation Paradigms," *Scientia Iranica 10(4),* pp. 383-391, 2003.

[9] Ghassem Jaberipur and Saeid Gorgin, "A Nonspeculative Maximally Redundant Signed Digit Adder," *The 13th international CSI Computer Conference,* pp. 235-242, 2008.

[10] John Moskal, Erdal Oruklu and Jafar Saniie, "Design and Synthesis of a Carry-Free Signed-Digit Decimal Adder," *IEEE International Symposium on Circuits and Systems,* pp. 1089-1092, 2007.