# Parallel and Digit-Serial Implementations of Area-Efficient 3-Operand Decimal Adders

**Tso-Bing Juang, Hsin-Hao Peng, Han-Lung Kuo**

***Abstract*—In this paper, parallel and digit-serial implementations of area-efficient 3-operand decimal adders are proposed. By using proposed analyzer circuits and the generation of correction terms with recursive schemes, our proposed decimal adders could perform efficient additions with three operands. Unit gate estimates and synthesis results show that our proposed adders are more area-efficient than those previously proposed decimal adders with three operands under the same delay constraints. Also the power consumptions for our decimal adders are lesser. In addition to parallel implementations, the digit-serial 3-operand adders are easily developed to increase the throughput and the operating frequency due to area efficiency. Our proposed decimal adders could be applied to ease the tremendous computation efforts for decimal computations such as multi-operand decimal additions, decimal multiplications and divisions.***

***Index Terms*—Computer arithmetic, Decimal additions, Parallel-prefix adders, VLSI design,**

## I. INTRODUCTION

Since the growth of decimal arithmetic in commercial, financial and internet-based applications, the use of hardware support for decimal arithmetic is becoming more and more important for the hardware designers and users. The decimal arithmetic is natural for human as we use ten fingers for counting numbers. In the past decades, although binary arithmetic is widespread used in the processors, there are some constraints in its use. For example, the binary numbers can not be used for the representations of some fractions, e.g., $0.3_{10}$=0.01001…..$_2$, which will require infinite bits for representation. This is not suitable for exact decimal fractions, since the incorrect results for the approximate representation of inputs will lead to subsequent approximation errors and thus will degrade the accuracy for the entire computations.

To remedy the drawback, the binary coded decimal (BCD) numbers is used as a common representation of decimal numbers, as BCD can recode each digit of decimal numbers from 0 to 9 using four bits $0000_2$ to $1001_2$, respectively. In the above example of the representation of $0.3_{10}$, the BCD numbers can only be recoded as 0.0011 (BCD) in finite and exact representations. Recently, the specifications for decimal floating-point arithmetic have been included in the draft of IEEE-754r standard for floating-point arithmetic [1].

**Manuscript received November, 2013.**

**Tso-Bing Juang** is with the Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce, Pingtung 900, Taiwan.

**Hsin-Hao Peng** is with the Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce, Pingtung 900, Taiwan.

**Han-Lung Kuo** is with the Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce, Pingtung 900, Taiwan.

Hence, designs of efficient decimal hardware are helpful in the operations of decimal numbers; and in the past the processors that include the compatibilities of IEEE-754r have been presented in the designs of IBM Power 6, z9 and z10 processors. However, the problems for using BCD numbers in decimal operations are the corrections of the digits when the range are greater than or equal to 10, the correction terms ($0110_2$) will be added to the digit and then producing the carries into the next digit so that may lead to long carry chain within the consecutive higher significant digits. In previously reported literature, the fast BCD adders with two and multi-operands are proposed in [2-11], the software implementations supported for IEEE-754r was proposed in [12], and the methods of fast BCD multiplications/divisions are presented in [13-24]. As for the survey of decimal units which can be referred in [25].

Our focus here is to design a decimal adder with three operands. Although this is a subset case of the multi-operand decimal additions which are already given in [5-6]. In this paper, we will point out under some conditions, the correction terms are required by using the 3-operand decimal adders proposed in [6], and thus the corrected implementation of [6] will be given. To achieve area-efficient implementation, our design is to improve the known fast BCD adders with two operands to perform the fast additions with three operands. The implementation of our proposed decimal adders provides a more economical way to achieve high-speed decimal addition with three operands and is very suitable for digit-serial implementation to increase the throughputs. By adopting proposed analyzer circuits and the generation of correction terms with recursive schemes, our proposed decimal adders could perform fast addition with three operands of four and eight digits with up to 67.4% area savings under the same delay constraints. The area overhead of our proposed adders is also lower than that of the corrected implementations of multi-operand decimal adders proposed in [6].

The rest of this paper is organized as follows. In Section II, we will introduce the designs of previous decimal adders with three operands. Then our proposed parallel and digit-serial implementations of area-efficient 3-operand decimal adders are given in Section III. The CMOS VLSI implementation results and comparisons will be presented in Section IV, and Section V concludes our work.

## II. PREVIOUS PROPOSED 3-OPERAND DECIMAL ADDERS

Before we introduce the designs of multi-operand decimal adders proposed in [5-6], we will describe two designs of previous decimal adders with two operands. Given two inputs of one digit BCD numbers $X$ and $Y$, the conventional architecture of the decimal addition of $X$ and $Y$ is depicted in Fig. 1, After using 1 digit adder composed of 4 consecutive

full adders to sum up the values of *X* and *Y*, another 1 digit adder is used to produce the decimal sum of *X* and *Y* (i.e., *Result*[3:0]) with the correction value 0110, which is determined by the output of $c \vee (S[3] \cdot S[2]) \vee (S[2] \cdot S[1])$, where $\vee$ and $\cdot$ are denoted as logical OR and logical AND operations, respectively, and *c* is the carry-out of the binary sum of *X* and *Y*.

We can observe that the delay of Fig. 1 will become longer as the digit-width of the decimal inputs increases since it will take the delay of 8 full adders and extra logic circuits for determining the correction term to be added with one-digit summation.

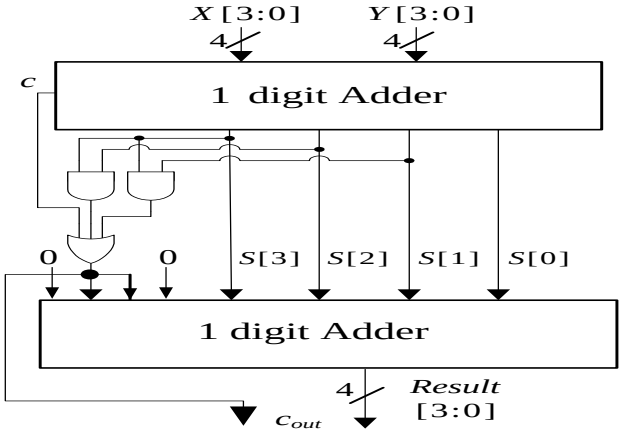


Fig. 1: Conventional hardware for performing one-digit BCD addition [26].

In [9], the authors proposed a method for fast BCD addition, which the adder is known as a fast decimal adder and was been slightly improved in [29] with almost the same delay and area complexity for fixing a few incorrect cases; hence we only describe the architecture in [9], which is depicted in Fig. 2 for four digit decimal addition with two operands *X*[15:0] and *Y*[15:0] and the carry-in $C_{in}$.

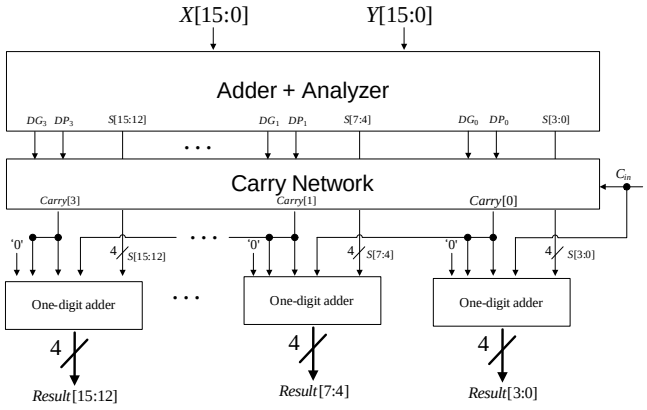


Fig. 2: The architecture of the reduced delay BCD adder proposed in [9].

In Fig. 2, it consists of three stages for fast BCD addition: in stage 1, two inputs are divided into four digits, and sent to the Adder+Analyzer hardware to produce the binary sums *S*[15:0] using four 4-bit carry lookahead adders (CLA). Also the Digit Propagate signals (i.e., $DP_i$) and the Digit Generate signals (i.e., $DG_i$) are produced for *i*=0 to 3. $DP_i$ and $DG_i$ signals are used to identify the conditions of the binary sums are equal to 9 and greater than 9, respectively.

$$\begin{cases} DG_i = C_{out} + (S[i+3] \cdot (S[i+2] + S[i+1])) \\ DP_i = S[i+3] \cdot S[i+1] \end{cases} \quad (1)$$

The logical expressions for $DP_i$ and $DG_i$ are denoted as Eq. (1), and the corresponding hardware is shown in Fig. 3. In stage 2, the signals $DP_i$ and $DG_i$ are sent to the Carry Network composed of parallel-prefix computation units to compute the real decimal carries *Carry*[i], i.e., $Carry[i]=DG_i+DP_i \bullet Carry[i-1]$. Then in the last stage, the correction values are parallel added to the binary sums produced by stage 1 to produce the real decimal sums *Results*[15:0] using four one-digit adders, and the carry-out for each one-digit adder can be discarded.

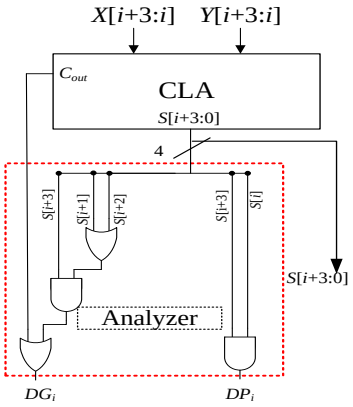


Fig. 3: The hardware of Adder+Analyzer for producing $DG_i$ and $DP_i$ proposed in [9].

Based on the methods in [26] and [9], the hardware for performing decimal addition with three 4-digit operands *X*, *Y* and *Z* can be developed straightforward. As the number of operands increases by 1, the hardware will be doubled. In addition, although the area consumption based on [26] is very lower compared to the one based on [9] for 3-operand decimal addition, the operation time for addition and correcting constitute to longer delays. On the other hand, the hardware based on [9] can perform parallel decimal additions, with larger area overhead than that of [26], which are composed of Analyzer and Carry Network.

In [6], high-speed multi-operand decimal adders were proposed. The authors proposed three architectures to speed-up the multi-operand decimal addition, including single correction, double correction, and nonspeculative ones. Among these, nonspeculative adder can achieve the highest speed, thus we only introduce the architecture of nonspeculative adder and use it with the ones based on [9] and [26] for comparison metrics in section 4.

Fig. 4 shows the architecture of one-digit, 3-operand nonspeculative adder. Using carry save adder (CSA) to produce the carry and sum vectors first, then the carry and sum vectors are summed using a 5-bit carry propagate adder accepting the previous most-significant bit of carry (i.e., $c'_1[3]$) to produce the intermediate sum $z'[4:0]$ and to be corrected with *g* using another 4-bit carry propagate adder (CPA) to obtain the final sum *Result*[3:0]. The most-significant bit of carry vectors (i.e., $c_1[3]$) is used with the values of $z'[3:0]$ to generate the 4-bit correction terms *g* and the carry-out (i.e., $c_{out}$) of the results using the circuit named as Sum and Carry Correction Logic, in which the corresponding function can be referred to [Table 4, 6]. According to [6], we have found that some incorrect digits will be produced in the final sum. Fig. 5 is the numerical example of 4-digit, 3 operand nonspeculative addition

adopted by [6]. We can observe that the final sum digits may be equal to or greater than 10; therefore, extra circuits for detecting incorrect digits, generation and the addition of correction terms (i.e., $0110_2$) which are similar to Fig. 1 are required to produce the final correct sums as shown in Fig. 6, leading to time- and area-consuming due to cascaded carry propagations, where FA denotes a full adder.

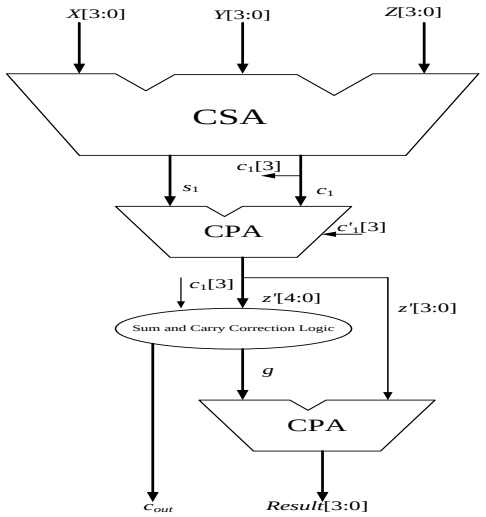


Fig. 4: One-digit, 3-operand nonspeculative adder proposed in [6].

| | | | | | |
|---|---|---|---|---|---|
| $X$= 5 9 4 1 = | | 0101 | 1001 | 0100 | 0001 |
| $Y$= 2 7 5 9 = | | 0010 | 0111 | 0101 | 1001 |
| $Z$= 0 3 0 0 = | | 0000 | 0011 | 0000 | 0000 |
| $S_1$ | = | 0111 | 1101 | 0001 | 1000 |
| $C_1$ | = | 00000 | 0110 | 1000 | 001 |
| $Z'$ | = | 00111 | 10011 | 01001 | 01010 |
| $G$ | = | 0000 | 0110 | 0000 | 0110 |
| $Z$ | = | 0111 | 1001 | 1001 | 0000 |
| $c_{out}$ | = 00 | 01 | 00 | 01 | |
| *Result* | 0 | 1000 | 1001 | 1010 | 0000 |
| | | 8 | 9 | 10 | 0 |

*Incorrect digit*

Fig. 5: Numerical example of incorrect result adopted by the method in [6].

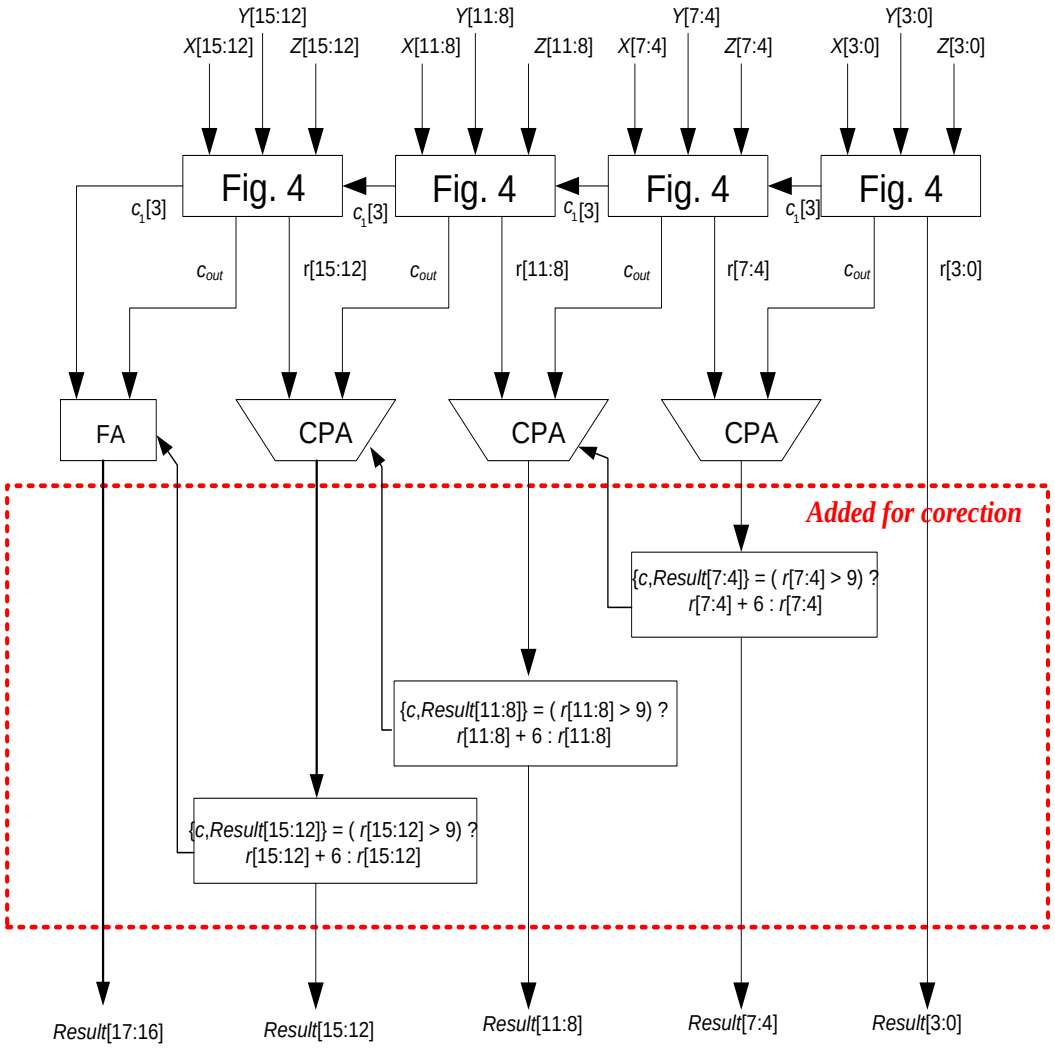


Fig. 6: The corrected architecture in [6] for nonspeculative decimal addition with three 4-digit operands.

In order to achieve area-efficient implementation, we improve the fast known BCD adders with two operands [9, 29] to perform fast decimal addition with three operands. In the next Section, we will propose the parallel and digit-serial implementations of area-efficient 3-operand decimal adders.

## III. PROPOSED AREA-EFFICIENT 3-OPERAND DECIMAL ADDERS

Similar to the method proposed in [9], our proposed parallel decimal adder shown in Fig. 7, which also consists of three stages. The first stage is to produce the binary sums and digit propagation and generation signals (i.e., *DP* and *DG*) using CSA+CLA+Analyzer circuit, then using a parallel-prefix Carry Network to generate the correction values for each digit. In the last stage the real decimal sums can be obtained by adding the binary sums and the correction values.

Fig. 8 is our proposed architecture of one-digit CSA+CLA+Analyzer circuit. Since there are three operands, we use one CSA and CLA to compute the sums $S[i+3:i]$ and the carry-out ($C_{out}$) first. The ranges of the sums are between 0 and 27, that is to say, the real decimal carry-outs may be 0, 1, and 2. In the design of Analyzer part, we use the following signals to indicate the conditions of sums as shown in Table 1: the digit generation signal (i.e., *DG*) is composed of 2-bit signals, which identifies if the sums are greater than 9 or 19 or not; and the digit propagation signal (i.e., *DP*) is composed of 4-bit signals, which identifies if the sums are equal to 8, 9, 18 and 19 or not, since the decimal carry-ins from the digits with lower weight may be 0, 1 and 2. It should be noted that each bit of *DG* or *DP* is exclusive with each other, respectively, and the corresponding logic expressions of *DG* and *DP* are given in Eq. (2).

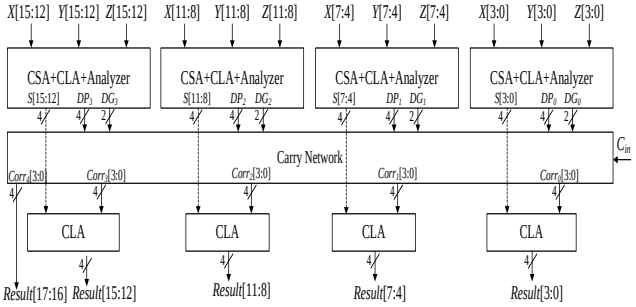


Fig. 7: The architecture of our proposed decimal adder with three 4-digit operands.

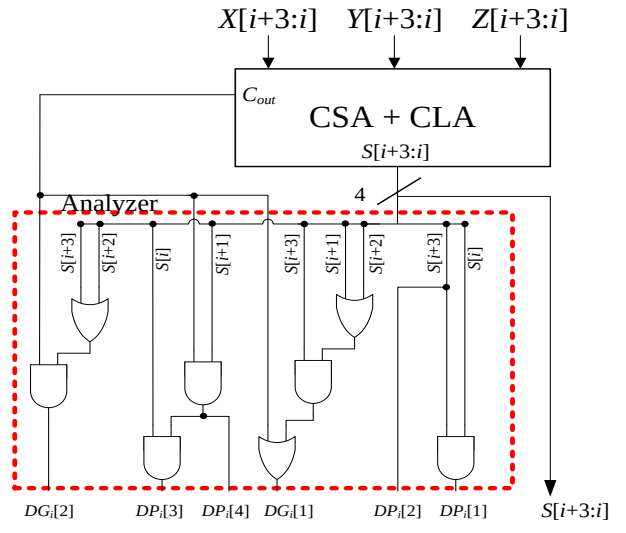


Fig. 8: One-digit hardware of our proposed CSA+CLA+Analyzer circuit.

TABLE I: DIGIT GENERATION AND PROPAGATION SIGNALS FOR IDENTIFYING THE CONDITIONS OF SUMS IN EACH DIGIT.

| Signals | Conditions of the sum in each digit |
|---|---|
| $DG_i[1]$ | >9 |
| $DG_i[2]$ | >19 |
| $DP_i[1]$ | =9 |
| $DP_i[2]$ | =8 |
| $DP_i[3]$ | =19 |
| $DP_i[4]$ | =18 |

$$\begin{cases} DG_i[1] = C_{out} + (S[i+3] \cdot S[i+2]) + (S[i+3] \cdot S[i+1]) \\ DG_i[2] = C_{out} \cdot (S[i+3] + S[i+2]) \\ DP_i[1] = DP_i[2] \cdot S[i+1] \\ DP_i[2] = S[i+3] \\ DP_i[3] = DP_i[4] \cdot S[i] \\ DP_i[4] = C_{out} \cdot S[i+1] \end{cases} \quad (2)$$

Now we define $C_{20}[i]$ and $C_{10}[i]$ as the values of carry-outs which are equal to 2 and 1 produced by i-th digit, respectively. The values of $C_{20}[i]$ and $C_{10}[i]$ can be obtained by Eq. (3) which is performed by the Carry Network stage shown in Fig.8:

$$\begin{cases} C_{10}[i] = DG_i[1] + DPi[1] \cdot C_{10}[i-1] + DP_i[2] \cdot C_{20}[i-1] \\ C_{20}[i] = DG_i[2] + DP_i[3] \cdot C_{10}[i-1] + DPi[4] \cdot C_{20}[i-1] \end{cases} \quad (3)$$

It can be easily seen that $C_{10}$ and $C_{20}$ can be computed by two any independent parallel-prefix computation unit simultaneously. After producing the values of $C_{10}$ and $C_{20}$ for each digit, the correction values denoted as $Corr_i[3:0]$ can be obtained by:

$$\begin{cases} Corr_i[0] = \overline{C_{20}[i-1]} \cdot C_{10}[i-1] \\ Corr_i[1] = (\overline{C_{20}[i]} \cdot C_{10}[i]) \oplus C_{20}[i-1] \\ Corr_i[2] = (\overline{C_{20}[i]} \cdot C_{10}[i] \cdot C_{20}[i-1]) \oplus (C_{10}[i] + C_{20}[i]) \\ Corr_i[3] = (\overline{C_{20}[i]} \cdot C_{10}[i] \cdot C_{20}[i-1]) \oplus C_{20}[i] \end{cases} \quad (4)$$

In the last stage, the correction stage, the binary sums of the first stage can be added by the correction values for each digit to produce the correct decimal sums. Since each carry-out produced by the sums of each digit of *X*, *Y* and *Z* is used to compute the digit propagation and generation signals; therefore, in the correction stage, the carry-outs produced by the sums and the correction values can be discarded, thus we can limit the carry propagation delay within one digit in the correction stage, and it won't cause longer carry propagation as [6] (corrected as shown in Fig. 6), [9] and [26]. Fig. 9 is the numerical example of our proposed decimal adders.

Table II summarizes the delay and area costs of our proposed and previous 3-operand *n*-digit decimal addition architectures, in equivalent gates, all assuming the unit gate model [27]. We assume the parallel-prefix adders used in [9] and our proposed method both follow the Brent-Kung architecture [28]. In order to produce the correct sums for the architecture proposed in [6], extra digit-adders shown in Fig. 6 are required for every digit except the least-significant digit; therefore, the delay and area costs will increase than the original one proposed in [6]. We can observe that the delay/area efficiency of our proposed method can outperform the ones proposed in [6] (corrected), [9] and [26].

In addition to parallel implementation of 3-operand decimal adders, we have also proposed the digital-serial adders as shown in Fig. 10, where *clk* denotes the clock cycle for completing the parallel implementation of *n*-digit 3-operand decimal addition. Since the area overhead for our proposed parallel design shown in Fig. 8 is much lower than that of previous proposed methods in [6] (corrected), [9] and [26], we can easily modify the parallel implementation into digit-serial ones to increase the throughput. In Fig. 10, the *n*-digit 3-operands are divided into *n* one-digits and sent into CSA+CLA+Analyzer from the least-significant digit first to the most-significant one. In every cycle, i.e., 1/*n clk*, the binary sum and the corresponding signals *DP* and *DG* are computed and output to the Carry Network to produce the correction terms with carry-in comes from the previous digit, and the carry-out and real sums are computed and stored in registers to be used in the next cycle. Since the Carry Network performs only 4-bit parallel-prefix computation, using CLA also has almost the same area and delay costs compared to using the parallel-prefix computation unit. As for the CMOS VLSI implementation results and comparisons will be given in the next Section.

TABLE II AREA AND DELAY COSTS ACCORDING TO THE UNIT-GATE MODEL

| Architecture | Delay | Area |
|---|---|---|
| [6] (corrected) | 61.5*n*-34.5 | 157*n*-31 |
| [9] | 4log$_2$*n*+14 | 128*n*-2log$_2$*n*-0.5 |
| [26] | 68*n*+2 | 118*n*+3.5 |
| **Proposed** | **2log$_2$*n*+15** | **4log$_2$*n*+88*n*-4** |

## IV. CMOS VLSI IMPLEMENTATION RESULTS AND COMPARISONS

All 3-operand decimal adders (including the corrected adder in [6] and the ones in [9], [26] and ours) were described in Verilog HDL and synthesized and mapped into a TSMC 0.18 μm CMOS standard-cell library using typical process parameters. The average dynamic power estimations for all adders were obtained by applying 1,000,000 random input vectors at a 250-MHz frequency at each design netlist and measured using Primepower. Table III and IV show the delay and area estimations for our proposed and previous adders in [6] (corrected and shown in Fig. 8), [9] and [26] with three 4-digit and 8-digit operands, respectively. The term N.A. denotes not available in the synthesis results. Table V shows the area and power consumptions under the minimum delays. According to Tables III and IV, we can observe that the area savings for our proposed adders can achieve up to 58.7% and 67.4% compared to the four-digit and eight-digit adders in [6], [9] and [26], respectively.

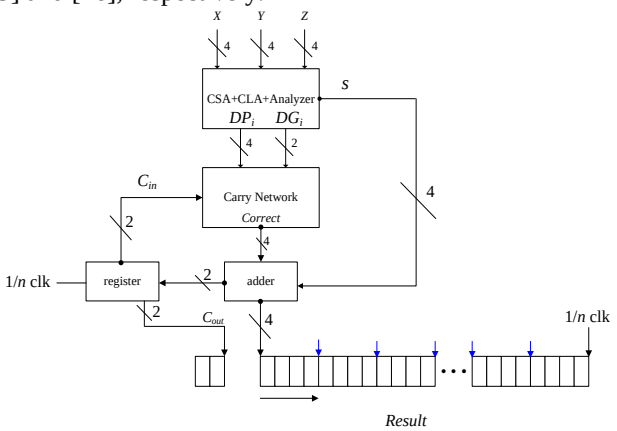


Fig 10: The architecture of our proposed digit-serial 3-oprand, *n*-digit decimal adder.

Compute :

| | |
|---|---|
| | $4675_{10}$ *X* |
| | $3856_{10}$ *Y* |
| + | $1992_{10}$ *Z* |
| | $10523_{10}$ |

**Adder and Analyzer**

| $C_{out3}$ | Digit-4 | $C_{out2}$ | Digit-3 | $C_{out1}$ | Digit-2 | $C_{out0}$ | Digit-1 | |
|---|---|---|---|---|---|---|---|---|
| | $0100_2$ | | $0110_2$ | | $0111_2$ | | $0101_2$ | *X* |
| | $0011_2$ | | $1000_2$ | | $0101_2$ | | $0110_2$ | *Y* |
| + | $0001_2$ | | $1001_2$ | | $1001_2$ | | $0010_2$ | *Z* |
| 0 | $1000_2$ | 1 | $0111_2$ | 1 | $0101_2$ | 0 | $1101_2$ | *Binary sum* |
| | 0 | | 1 | | 1 | | 1 | *DG*[1] |
| | 0 | | 1 | | 1 | | 0 | *DG*[2] |
| | 0 | | 0 | | 1 | | 1 | *DP*[1] |
| | 1 | | 0 | | 1 | | 1 | *DP*[2] |
| | 0 | | 1 | | 0 | | 0 | *DP*[3] |
| | 0 | | 1 | | 0 | | 0 | *DP*[4] |

**CarryNetwork**

| $C_{out}$ | Digit-4 | Digit-3 | Digit-2 | Digit-1 | |
|---|---|---|---|---|---|
| | 0 | 1 | 1 | 1 | *DG*[1] |
| | 0 | 1 | 1 | 0 | *DG*[2] |
| | 0 | 0 | 1 | 1 | *DP*[1] |
| | 1 | 0 | 1 | 1 | *DP*[2] |
| | 0 | 1 | 0 | 0 | *DP*[3] |
| | 0 | 1 | 0 | 0 | *DP*[4] |
| 1 | 1 | 1 | 1 | 1 | $C_{10}$ |
| 0 | 0 | 1 | 1 | 0 | $C_{20}$ |
| $0001_2$ | $1000_2$ | $1110_2$ | $1101_2$ | $0110_2$ | *Corr* |

**CLA**

| $C_{out}$ | Digit-4 | Digit-3 | Digit-2 | Digit-1 | |
|---|---|---|---|---|---|
| | $1000_2$ | $0111_2$ | $0101_2$ | $1101_2$ | *Binary sum* |
| + $0001_2$ | $1000_2$ | $1110_2$ | $1101_2$ | $0110_2$ | *Corr* |
| $0001_2$ | $0000_2$ | $0101_2$ | $0010_2$ | $0011_2$ | *Result* |

Fig. 9: Numerical example of our proposed methods.

**According to Table** V, the minimum delay of our proposed adders with three operands for four and eight digits is 2.9 ns and 3.9 ns, respectively, which can outperform previous adders proposed in the corrected ones in [6] (4.3 ns for *n*=4 and 8.9 ns for *n*=8), [9] (3.5 ns for *n*=4 and 4.1 ns for *n*=8) and [26] (7.4 ns for *n*=4 and 12.5 ns for *n*=8).

Also we can obtain that our proposed adders could achieve up to 86.8% and 90.8% ADP (Area×Delay×Power) product savings over previous adders under *n*=4 and 8, respectively. The reasons for the ADP product efficiency achieved by our proposed adders are that we use two independent parallel-prefix computation units to compute the correction terms at the same time, and we limit the carry propagation delay of final correcting into one digit only. The minimum delay and the corresponding area (including extra registers) for our proposed digit-serial adder are 1.5 ns and 3,153 μm$^2$, respectively. In other words, the working frequency of our proposed digit-serial adder could be up to 666 MHz.

TABLE III AREA COMPARISONS UNDER DIFFERENT DELAY CONSTRAINTS (3-OPERAND, 4-DIGIT ADDERS)

| Delay (ns) | Area (μm$^2$) | | | | Area savings over (%) | | |
|---|---|---|---|---|---|---|---|
| | [6] (corrected) | [9] | [26] | Ours | [6] (corrected) | [9] | [26] |
| 7.4 | 9869 | 4960 | 8116 | 4071 | 58.7 | 17.9 | 49.8 |
| 3.7 | N. A. | 9862 | N.A. | 4667 | N.A | 52.6 | N.A. |
| 2.9 | N. A | N. A. | N.A | 7158 | N.A | N.A | N.A |

TABLE IV AREA COMPARISONS UNDER DIFFERENT DELAY CONSTRAINTS (3-OPERAND, 8-DIGIT ADDERS)

| Delay (ns) | Area (μm$^2$) | | | | Area savings over (%) | | |
|---|---|---|---|---|---|---|---|
| | [6] (corrected) | [9] | [26] | Ours | [6] (corrected) | [9] | [26] |
| 13.1 | 24239 | 9407 | 17051 | 7903 | 67.4 | 15.9 | 56.6 |
| 4.1 | N. A. | 19702 | N. A. | 12067 | N. A. | 38.7 | N. A. |
| 3.9 | N. A. | N. A. | N. A. | 14536 | N. A. | N. A. | N. A. |

TABLE V AREA AND POWER CONSUMPTIONS UNDER THE MINIMUM DELAYS

| Architectures | Area(μm$^2$) | Delay(ns) | Power (mW) |
|---|---|---|---|

| | *n*=4 | *n*=8 | *n*=4 | *n*=8 | *n*=4 | *n*=8 |
|---|---|---|---|---|---|---|
| [6] (corrected) | 19273 | 38233 | 4.3 | 8.9 | 1.48 | 3.33 |
| [9] | 10123 | 19702 | 3.5 | 4.1 | 1.67 | 2.95 |
| [26] | 8432 | 19331 | 7.4 | 12.5 | 1.23 | 2.99 |
| Ours | 7158 | 14536 | 2.9 | 3.9 | 0.78 | 1.84 |

## V.CONCLUSIONS

We have proposed parallel and digit-serial implementations of area-efficient 3-operand decimal adders. By using proposed analyzer circuits and the generation of correction terms with recursive schemes, our proposed decimal adders could perform fast addition of three operands with up to 67.4% area savings under the same delay constraints. Also the corrected hardware for multi-operand decimal adders in [6] is given. The proposed decimal adders could be applied to ease the tremendous computation efforts for decimal computations such as multi-operand decimal additions, decimal multiplications and divisions.

## REFERENCES


[1] Draft IEEE Standard for floating-point arithmetic. New York: IEEE, Inc., 2004, http://754r.ucbtest.org/drafts.

[2] M. S. Schmookler and A.W.Weinberger. "High speed decimal addition," *IEEE Transactions on Computers*, Vol. 20, No. 8, pp. 862–867, Aug. 1971.

[3] M. A. Erle, M. J. Schulte, and J. M. Linebarger, "Potential speedup using decimal floating-point hardware," *Proc. of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, Vol. 2, pp. 1073–1077, Nov. 2002.

[4] M. F. Cowlishaw, "Decimal floating-point: algorism for computers," *Proc. of 16th IEEE Symposium on Computer Arithmetic* (*ARITH-16*), pp. 104–111, June 2003.

[5] R.D. Kenney and M.J. Schulte, "Multioperand decimal addition," *Proc. IEEE Computer Society Ann. Symp. VLSI*, pp. 251-253, Feb. 2004.

[6] R.D. Kenney and M.J. Schulte, "High-speed multioperand decimal adders," *IEEE Transactions on Computers*, pp. 953-963, Vol. 54, No. 8, Aug. 2005.

[7] Thapliyal, H, Kotiyal. S, Srinivas, M.B., "Novel BCD adders and their reversible logic implementation for IEEE 754r format", *Proc. 19th IEEE International Conference on VLSI Design*, pp. 3-7, Jan. 2006

[8] Sreehari Veeramachaneni, M.Kirthi Krishna, Lingamneni Avinash, Sreekanth Reddy P, M.B. Srinivas, "Novel, high-speed 16-digit BCD adders conforming to IEEE 754r format," *Proc. IEEE Computer Society Ann. Symp. VLSI* (*ISVLSI*'07), pp. 343-350, May 2007.

[9] A. Bayrakci and A. Akkas, "Reduced delay BCD adder," *Proc. IEEE 18th International Conference on Application-specific Systems, Architectures and Processors, (ASAP)*, pp. 266-271, July 2007.

[10] G. Bioul, M. Vazquez, J. P. Deschamps, and G. Sutter, "Decimal addition in FPGA," *Proc. SPL. 5th Southern Conference on Programmable Logic*, pp. 101-108, 2009.

[11] A. Vazquez and E. Antelo, "A high-performance significand BCD adder with IEEE 754-2008 decimal rounding," *Proc. 19th IEEE Symposium on Computer Arithmetic (ARITH-19)*, pp. 135-144, 2009.

[12] E. Cornea, J. Harrison, J. C. Anderson, P. Tang, E. Schneider, and E. Gvozdev, "A software implementation of the IEEE 754r decimal floating-point arithmetic using the binary encoding format," *IEEE Transactions on Computers*, Vol. 58, No. 2, pp. 148-162, Feb. 2009.

[13] M.A. Erle and M.J. Schulte, "Decimal multiplication via carry-save aAddition," *Proc. IEEE 14th Int'l Conf. Application-Specific Systems, Architectures, and Processors*, pp. 348-358, June 2003.

[14] M. A. Erle, E. M. Schwarz, and M. J. Schulte, "Decimal multiplication with efficient partial product generation," *Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, pp. 21-28, 2005.

[15] M. A. Erle, M. J. Schulte and B. J. Hickmann, "Decimal floating-point multiplication via carry-save addition," *Proc. 18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pp. 46-55, 2007.

[16] B. J. Hickmann, A. Krioukov, A.M. J. Schulte and M. A. Erle, "A parallel IEEE P754 decimal floating-point multiplier," *Proc. 25th International Conference on Computer Design (ICCD)*, pp. 296-303, 2007.

[17] G. Jaberipur, and A. Kaivani, "Binary-coded decimal digit multipliers," *IET Computers & Digital Techniques*, Vo1. 1, No. 4, pp. 377-381, 2007.

[18] A. Vazquez, E. Antelo and P. Montuschi, "A new family of high performance parallel decimal multipliers," *Proc. of the 18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pp. 195-204, 2007.

[19] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "Decimal multiplication using compact BCD multiplier," *Proc. International Conference on Electronic Design (ICED)*, pp. 1-6, 2008.

[20] G. Jaberipur and A. Kaivani, "Improving the speed of parallel decimal multiplication," *IEEE Transactions on Computers*, Vol. 58, No. 11, pp. 1539-1552, Nov. 2009.

[21] A. Vazquez, E. Antelo and P. Montuschi, "Improved design of high-performance parallel decimal multipliers," *IEEE Transactions on Computers*, Vol. 59, No. 5, pp. 679-693, May 2010.

[22] H. Nikmehr, B. Phillips, and C. -C. Lim, "Fast decimal floating-point division," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 14, No. 9, pp. 951-961, Sept. 2006.

[23] T. Lang, and A. Nannarelli, "Division unit for binary integer decimals," *Proc. 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1-7, 2009.

[24] T. Lang and A. Nannarelli, "A radix-10 digit-recurrence division unit: algorithm and architecture," *IEEE Transactions on Computers*, Vol. 56, No. 6, pp. 727-739, June 2007.

[25] L. -K. Wang, M. A. Erle, C. Tsen, E. M. Schwarz and M. J. Schulte, "A survey of hardware designs for decimal arithmetic," *IBM Journal of Research and Development*, Vol. 54, Issue 2, pp. 8:1-8:15, 2010.

[26] P. Parhami, Computer arithmetic: algorithms and hardware designs. New York: Oxford Univ. Press, 2000.

[27] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Transactions on Computers*, Vol. 42, No. 10, pp. 1163–1170, Oct. 1993.

[28] R. Brent and H. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, Vol. 31, No. 3, pp. 260-264, 1982.

[29] C. Sundaresan, CVS Chaitanya, PR Venkateswaran, Bhat Somashekara, and J. Mohan Kumar, "Modified reduced delay BCD adder," *Proc. 4th International Conference on Biomedical Engineering and Informatics* pp. 2148-2151, 2011.



**Tso-Bing Juang** received the PhD degree of the Computer Science and Engineering from National Sun Yat-Sen University, Taiwan, in Dec. 2004. He served as a lecturer and assistant professor at Ta Jen University during Aug. 1998 to Jan. 2006 prior to joining the department of Computer Science and Information, National Pingtung Institute of Commerce (NPIC), Taiwan in Feb. 2006, where now he is currently an Associate Professor since Aug. 2010. He has received the Best Tutoring Award and Outstanding Teaching Award by NPIC in 2010 and 2011, respectively. Since Dec. 2011, he was the visiting researcher at the Centre for High-Performance Embedded Systems (CHiPES), Nanyang Technological University (NTU) in Singapore under financial support by National Science Council in Taiwan.

Dr. Juang was a receiver of Xerox Best Thesis Award in Taiwan for the contributions of his master thesis in 1995. He has co-authored one textbook about full-custom IC design (in Chinese) and authored more than 40 research papers in referred international and domestic journals and conferences. He was the principal investigator and the co-investigator of 7 research projects supported by National Science Council in Taiwan since 2007. Also he was the principal investigator of 3 industry-academy research projects supported by the Ministry of Education in Taiwan, in 2007, 2008 and 2009, respectively. He served as the peer reviewers for IEEE journals such as IEEE Transactions on Circuits and Systems (I) and (II), IEEE Transactions on Computers, and IEEE Transactions on VLSI Systems.

**Hsin-Hao Peng** received his Master's degree from the Department of Computer Science and Information at National Pingtung Institute of Commerce, Taiwan, in 2011.

**Han-Lung Kuo** received his Master's degree from the Department of Computer Science and Information at National Pingtung Institute of Commerce, Taiwan, in 2012.