Volume 9, Issue 4
August 2020

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Designing an On-Line Magnitude Comparator for Higher-Radix

Dr. Madhu Sudan Chakraborty

Department of Computer Science, Indas Mahavidyalya, Indas, Bankura (WB), India, Pin 722205

E-mail: mailmschakraborty@rediffmail.com

ABSTRACT

On-line arithmetic attracts the computer arithmetic community primarily owing to its ability to admit digit-level pipelining of several operations, apparently having distinct computational characteristics. The pipelining results in substantial reduction in the number of clock cycles needed for computing and consequently ensures better performance of the processor. The magnitude comparator is an indispensable part of any arithmetic processor. Magnitude comparator seemingly involves complex design, whatsoever number system, conventional or unconventional, is employed. However, in this paper, it is strived to be shown that in on-line arithmetic platform a simple magnitude comparator for higher-radix may be realized at ease. It may be done thanks to interpreting magnitude comparison in terms of additions/subtractions and sign-detections and the ability to circulate all resulting digits from the most-significant position to the least-significant position in an effective and efficient manner. Thus it may be possible to proceed further towards developing a full-fledged processor using on-line arithmetic only.

Keywords: On-line arithmetic, magnitude comparator, higher-radix, ordinary signed-digit number systems, sign-detection

1. INTRODUCTION

On-line arithmetic (OLA), introduced in [1], reflects a new trend in computer arithmetic (CA) where the digits are passed from the most-significant (MS) to least-significant (LS) position ([2]-[3]). In OLA an output digit of an operation may be consumed by the next operator as an input before all remaining output digits of the former operation are produced. Thus various operations (like addition/subtraction, multiplication, division) may be arranged to overlap, causing a digit-level pipelining and consequently substantial reduction in the number of computational clock cycles. As a result the processor may perform better. The number system (NS) employed by any OLA unit is necessarily a redundant NS like signed-digit number system (SDNS) [4]-[7]. In OLA platform, some operations have been found to be efficiently implementable [1]-[3], [8]-[9] but no report is available yet for magnitude comparison (MCN) which is also an important arithmetic operation (AO). It is mentionable that for any CA unit employing whatsoever NS, conventional or unconventional, MCN seems to be a complex operation and investigations to develop better magnitude comparators (MCRs) is being continued [10]-[12].

In this paper, the author would like to investigate designing an on-line magnitude comparator (OLMCR) using typical signed-digit arithmetic (SDA), particularly thanks to its absolute/ almost carry-free, simple addition/ subtraction property and the technique of interpreting MCN in terms of addition/ subtraction and sign-detection (SDTN) as presented in [12]. The proposed CA algorithm (CAA) obviously deals with higher-radix, in line with the observation that high-radix OLA may lead to more credible and accurate computing [13].

The rest portion of the paper is organized with four sections as follows: In the Related Works section the discussion starts with specifying the current state of the MCN problem, followed by stating the key issues of OLA and SDNS in brief and thereafter the schematic realization of on-line adder for higher-radix and an OLA-algorithm (OLAA) for SDTN are presented. In Designing On-line Comparator for Higher-Radix section an OLMCR is strived to be developed for higher radix and the schematic diagram of the proposed algorithm is also shown. In the Results and Discussion section the step-by-step operations of the proposed algorithm are shown with an example. Finally the study is ended in the Conclusion section, indicating its major contribution to the literature.

2. RELATED WORKS

2.1. MAGNITUDE COMPARISON

Let A and B are two n-digit unsigned binary numbers where $A = A_{n-1}A_{n-2}........A_0$ and $B = B_{n-1}B_{n-2}........B_0$. The ordinary pencil-and-paper method (PPM) for MCN of A and B is presented as algorithm 1 as follows:

```
Algorithm 1:
  a.  Initialize: $i = n-1$
  b.  If ($A_i>B_i$) then A>B.
      Otherwise, if ($A_i<B_i$) then A<B
      Exit
  c.  If ($A_i=B_i$) then
        i.  If ($i=0$) then A=B
            Exit
        ii. Else set: $i=i-1$
            Go to step (b)
```

Although looking simple to implement, PPM for MCN is too slow owing to linear dependency of the iterative values and obviously it consumes O(n) time.

Recently a faster MCN algorithm has been developed on the basis of divide-and-conquer philosophy. Let $A=A_HA_L$ and $B=B_HB_L$ where A_H and B_H represent the most-significant half (MSH) of A and B respectively and A_L and B_L represent the least-significant half (LSH) of A and B respectively. Then A>B if (A_H>B_H) or ((A_H=B_H) and (A_L>B_L)). Alternatively A<B if (A_H<B_H) or ((A_H=B_H) and (A_L<B_L)). Also A=B if ((A_H=B_H) and (A_L=B_L)). Then using bottom-up approach a O(logn) time MCR for A and B may be easily developed [10]. A sample execution sequence for an ordinary log-depth MCN scheme dealing with unsigned numbers is shown in Fig. 1 as an example where A=10110011 and B=11000010. In Fig. 1 L1, L2, L3 and L4 denote the level 1, level 2, level 3 and level 4 respectively.

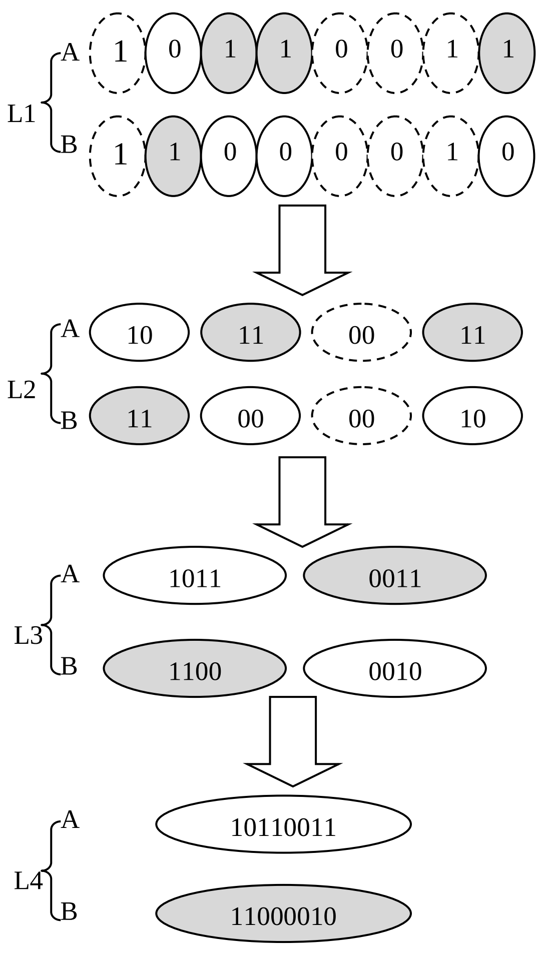


Fig. 1: Log-depth MCN Approach

As shown in Fig. 1 for the same level the comparison involves the (vertically) upper and lower cells and the result is expressed by either shading or fixing boundary-characteristics of the concerned cells. Any shaded cell has higher magnitude than its un-shaded counterpart. On the other hand, the cells marked by dotted boundaries have the equal magnitude.
In this sub-section discussion held so far has been concerned with the conventional NS (CNS). For the unconventional NS (UCNS) designing a MCR is also a challenging task [5], [11]-[12].

2.2. ON-LINE ARITHMETIC

In the conventional radix-complement arithmetic, for some operations (like division and square root) the results are generated from MS-digit (MSD) to LS-digit (LSD) and for some other operations (like addition and multiplication) the results are produced from LSD to MSD. For arithmetic-intensive applications data need to propagate among many different operations and so large latency may occur. An alternative idea would be to explore the scope to generate all resulting digits of every AO to a particular direction only.

The digit-serial arithmetic [6] seems to match to the idea and it may perform better than the ordinary fixed-point arithmetic. There exist two different versions of digit-serial arithmetic: LSD-first (LSDF) and MSD-first (MSDF). In the literature MSDF arithmetic has been termed as OLA and it appears to be more effective and efficient compared to LSDF arithmetic. OLA has been being intensively investigated over the fields of ASIC and DSP. In digit-serial OLA ɤ+1 digits of the input operand are required for computing the first digit of the result and after that, for each new digit of the input operand, an extra digit of the result is produced. In this connection, ɤ is called the online delay. Computing online delay for a typical problem is shown in Figure 2.

Cycle	$\bar{2}$	$\bar{1}$	0	1	2	.....
Input	X_1	X_2	X_3	X_4	X_5	.....
Compute	...	....	....	....	....	.....
Output				Z_1	Z_2	

So ɤ=2

Figure 2: Computing Delay in OLA

Although originally being introduced as digit-serial arithmetic, later the parallel version of OLA, called parallel OLA, has been introduced too [6]. In parallel OLA the scope of intra-operation parallelism (like parallel addition) is explored too. Obviously the parallel OLA may decrease latency at the cost of employing addition hardware ([14], [15]). In article [16] efficient FPGA implementation of online adders and multipliers are discussed. In article [17] it is observed that accumulation of timing errors may be resisted thanks to the MSDF computing observed in OLA and obviously timing errors may appear at LSDs only. However, the consumption of more computing resources compared to the traditional fixed-point arithmetic circuits appears to be a drawback of the parallel OLA driven arithmetic circuits.

On the other hand, digit-serial OLA involves small circuitry and consumes less power, still resulting in high-speed and so it is suited for real-time control systems ([18]-[19]). However, serial OLA usually consume more clock cycles than the parallel. So the decision on using a particular mode of OLA (serial/ parallel) ultimately rests with the desired application characteristics. In this work henceforth employing serial OLA is assumed, unless explicitly stated.

Recently the implementation issues of the common AOs (namely, addition, subtraction, multiplication and division) in OLA mode have been investigated in details in [8]. Another major contribution of the paper [8] is its ability to precisely regulate the numerical precision of addition during the run-time, unlike the ordinary OLA [2] where precision is to be fixed at design time. The scheme proposed in [8] exhausts the fixed-precision adder and also employs on-chip RAM for storing the residues, resulting in a variable precision adder.

2.3. THE SIGNED-DIGIT NUMBER SYSTEMS

For MSDF computation some redundant NS (in form of SDNSs) or redundant representation (like carry-save) needs to be employed. This is because, owing to the inherent redundancy in this case, computing the result in MSDF mode may be possible merely on the basis of the partial information of the inputs, following some refinements after the arrival of the other (or actual) inputs. In this regard, SDNSs seem to be more promising ([1]-[3]). The SDNS is a positional NS but yet it is unconventional in the sense that each digit of its digit set (DS) carries its independent sign. The most primitive form of SDNS is the ordinary SDNS (OSDNS). A radix-r OSDNS is defined as an unconventional, positional NS that works on the DS $\{\bar{\alpha}, \bar{\alpha}+1, .., \bar{1}, 0, 1, .., \alpha-1, \alpha\}$ where r/2 < α < r [4]. The DS of the radix-4 OSDNS is, for example, given by $\{\bar{3},\bar{2},\bar{1},0,1,2,3\}$. The most important contribution of the SDA is allowing addition/ subtraction to be performed in a time which is precision-independent (or constant-time). The constant-time addition/subtraction algorithm for OSDNS, Algorithm 2, is as follows:

Algorithm 2

```
a)  Input: two n-digit OSDNs as: $X=(X_{n-1}X_{n-2}\ldots X_0)$ and $Y=(Y_{n-1}Y_{n-2}\ldots Y_0)$.
b)  An intermediate sum, sum-1, $P=(P_{n-1}P_{n-2}\ldots P_0)$ is computed as: $P_i = X_i + Y_i \;\forall\, i \in [0, n-1]$
c)  Another intermediate sum, sum-2, $S'=(S'_nS'_{n-1}S'_{n-2}\ldots S'_0)$ and a carry $C=(C_nC_{n-1}\ldots C_1)$ is computed as:
    $S'_i = P_i - r.C_i \;\forall\, i \in [0, n-1]$
    Where $C_i = \begin{cases} \bar{1} & \text{if } P_i \le \bar{\alpha} \\ 0 & \text{if } \bar{\alpha} < P_i < \alpha \\ 1 & \text{if } P_i \ge \alpha \end{cases}$
    Initially $C_0 = 0$ and lastly $S'_n = 0$
d)  The final sum $S = (S_nS_{n-1}\ldots S_0)$ is computed in terms of:
    $S_{i+1} = S'_{i+1} + C_i \;\forall\, i \in [0, n-1]$
```

Here S_n is the nth (extra) digit used for correctly accommodating the sum. Stepwise execution of the constant-time addition using Algorithm 2 is shown in Fig. 3 as an example where X = $(\bar{3}\bar{1}10\bar{2}3)_4$ and Y = $(\bar{1}3320\bar{2})_4$

X		$\bar{3}$	$\bar{1}$	1	0	$\bar{2}$	3
Y		$\bar{1}$	3	3	2	0	$\bar{2}$
P		$\bar{4}$	2	4	2	$\bar{2}$	1
S′		0	2	0	2	$\bar{2}$	1
C	$\bar{1}$	0	1	0	0	0	×
S	$\bar{1}$	0	3	0	2	$\bar{2}$	1

Fig. 3: Parallel Addition in OSDNS: An Example

Here X = $-(3269)_{10}$, Y = $-(34)_{10}$ and S = $-(3303)_{10}$. Obviously S=X+Y holds true. Constant-time addition/subtraction is a significant achievement in computing as adders are often viewed as the basic building block of any arithmetic unit (AU) and consequently even some more complex operations, like multiplication and division, may run faster. Thus SDNSs seem suitable for digital signal processing, image processing and cryptography.

Even after the OSDNS many other SDNSs have come into existence with broader areas of applications. Generalized SDNS (GSDNS) [7] may be viewed as a super-class of OSDNS. For GSDNS DS = $\{\bar{\alpha}, \bar{\alpha}+1, .., \bar{1}, 0, 1, .., \beta-1, \beta\}$ where α ≥ 0, β ≥ 0 and α + β + 1 > r. The redundancy index of GSDNS is defined as: $\rho = \alpha + \beta + 1 - r$. Obviously the radix-4 SDNS with DS = $\{\bar{3}, \bar{2} \ldots, 0, 1, ., 3\}$ is more redundant compared to the radix-4 SDNS with DS = $\{\bar{2}, \bar{1}, \ldots, 0, 1, ., 2\}$. A class of SDNS, called binary SDNS (BSDNS) works on the DS $\{\bar{1},0,1\}$ and it seems to be one of the widely investigated classes of SDNDS. Recently another class of SDNS, called canonical SDNS (CSDNS) has got a lot press [5].

For realizing the SDNS in hardware some mappings from its unconventional digits to binary strings are needed. For BSDNS two widely used encoding techniques are positive-negative encoding (PNE) and two's-complement encoding (TCE) ([5], [20]). In PNE the binary signed-digits (BSDs) $\bar{1}$, 0 and 1 are encoded as 01, 00 and 10 respectively. On the other hand, in TCE the BSDs $\bar{1}$, 0 and 1 are encoded as 11, 00 and 01 respectively. For the higher-radix SDNSs (HRSDNSs) TCE may be extended. For instance, for radix-4 OSDNS the signed-digits (SDs) $\bar{3}, \bar{2}, \bar{1}, 0, 1, 2, 3$ of its DS may be encoded as 101, 110, 111, 000, 001, 010, 011 respectively.

Some other notable features of SDNSs include regularity in designing arithmetic circuit, low power consumption for performing some operations and fault-tolerance.

SDNSs offer regularity owing to its ability to perform the different stages of any multi-stage operation using the similar-looking arithmetic equations (AEs) and obviously employing look-alike hardware [5].

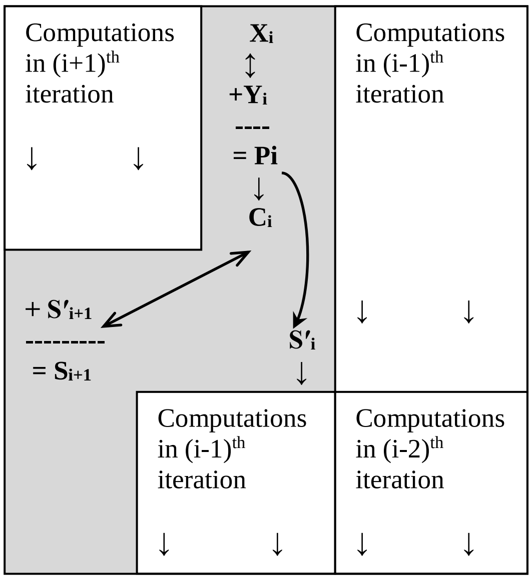
Recently the demand for low-power consuming arithmetic circuits has increased drastically. This is not only to support the increasing fabrication density of the IC, but also ensuring greater portability. However, speed and power often appear as two conflicting parameters and a trade-off needs to be maintained. Even though SDNSs are basically introduced for high-speed, it may be able to support low-power computing too. Research has shown that binary signed-digit (BSD) adder may consume less power compared to the traditional two's-complement (TC)-adder [21]. In the literature, employing SDA some other AOs consuming low-power are also known, including addition, multiplication and division ([22]-[24]).

For SDNSs the carry/borrow propagation is restricted and so the effect of any fault at some positional digit tends to be localized, leading to achieve more fault-tolerance in AUs [25].

In the primitive form SDNSs do not support the floating-point computations. However, unless addressing floating-point issues no stand-alone processor may be designed. Recently some variants of SDNSs have been introduced for allowing floating-point computations ([26]-[27]).

2.4. REALIZING ON-LINE ADDER FOR HIGHER-RADIX

The problem of designing on-line adder, particularly, may be solved employing OSDNSs at ease. This is because, in that case as shown in Algorithm 2, the sum-digit resulted at ith iteration is refined and settled at (i+1)th iteration using the incoming carry available at that iteration.


Computations in (i+1)th iteration
X_i
$+Y_i$

$= P_i$
C_i
Computations in (i-1)th iteration
$+ S'_{i+1}$

$= S_{i+1}$
S'_i
Computations in (i-1)th iteration
Computations in (i-2)th iteration

Figure 4: An OLA Adder for Higher-Radix

2.5. REALIZING ON-LINE SIGN-DETECTOR

Obviously no further revision is required during the subsequent iterations. The computational delay of a n-digit online adder is given by the delay caused by (n+2) full adders. The schematic diagram for ith iteration of an OLA adder for radix>2 is shown is Fig. 4.

Sign-detector may act as the kernel of some AUs employing SDNSs [28]. The sign of a sign-digit number (SDN) is evident from the sign of its MS non-zero digit. Unlike to the complex SDTN process in the traditional SDA [29], OLA provides a simple, straightforward solution of the SDTN problem. For a radix-r SDN $A=A_{n-1}A_{n-2}.......A_0$, an OLAA for SDTN, Algorithm 3, is presented as follows:

```
Algorithm 3
a)  Initialize: i=n-1, PA=0
b)  Set: $E_i$ = sign ($A_i$)
c)  Compute: PA = PA ø $E_i$
    Where X ø Y represents sign of X if $X\neq 0$ and otherwise it
    represents the sign of Y.
d)  Do:
     i.   Set: i=i-1
     ii.  If ($i\geq 0$) Go to step (b)
     iii, Otherwise,
              Output: PA
              Exit
```

3. DESIGNING ON-LINE COMPARATOR FOR HIGHER-RADIX

For any two (signed or unsigned) numbers, X and Y, as $X^2 - Y^2 = (X+Y). (X-Y)$, if the sign of either (X+Y) or (X-Y) is zero then $|X|=|Y|$. Otherwise, if the signs of (X+Y) and (X-Y) agree then $|X|>|Y|$ holds true and if the signs of (X+Y) and (X-Y) disagree then $|X|<|Y|$ holds true. Thus for any NS, MCN may be interpreted in terms of additions/subtractions and SDTNs and some comparator may be designed in a straightforward manner. Further thanks to the constant-time addition/ subtraction property of SDNS, the time complexity characteristics of both MCN and SDTN are the same [12]. As shown in sub-section 2.4 in OSDNS for any two signed-digit number (SDN)-input the sum/difference and carry-out/ borrow-out at/from any position during addition/ subtraction depends only the immediate lower-significant digits, besides those particular positional digits and so digit-serial output may be efficiently generated from MS-to-LS position. In addition SDTN may be circulated digit-serially from MS-to-LS position as presented in the sub-section 2.4. Thus proceedings towards a simple OLMCR design may be possible with efficacy as well as efficiency.

In the regard, for processing the following notations are used with respect to ith position $\forall$ i ϵ [-1, n] where n is the number of input digits:

P_i means pre-intermediate sum
Q_i means pre-intermediate difference
C_i means carry-out
B_i means borrow-out
S'_i means intermediate sum
D'_i means intermediate difference
S_i means actual sum
D_i means actual difference
E_i sign-information of S_i
F_i sign-information of D_i
PE means sign-information of SDN $S_nS_{n-1}....S_{i+1}$
PF means sign-information of SDN $D_nD_{n-1}....D_{i+1}$

Where all of E_i, F_i, PE and PF are assumed to represent positive, zero and negative signs by 1, 0 and $\bar{1}$ respectively, employing some standard 2-bit encoding as discussed in sub-section 2.3. Consider an OSDNS defined on radix r (>2) and DS $\{\bar{\alpha}, \bar{\alpha}+1, \ldots\ldots\ldots, \bar{1}, 0, 1, \ldots\ldots\ldots, \alpha-1, \alpha\}$ where $\frac{r}{2} < \alpha < r$. Let $X = X_{n-1}X_{n-2}\ldots\ldots\ldots X_0$ and $Y = Y_{n-1}Y_{n-2}\ldots\ldots\ldots Y_0$ be two numbers of the NS. Then an OLMC-algorithm, algorithm 4, is proposed as follows:

Algorithm 4:

```
a. Set: $X_{-1} = 0$, $Y_{-1} = 0$, $S'_n = 0$, $D'_n = 0$, PE = 0 and PF = 0
b. For i = n–1 down to $\bar{1}$ compute step i through vii:
    i. $P_i = X_i + Y_i$, $Q_i = X_i + \bar{Y_i}$
    ii. $C_i = \begin{cases} 1, & \text{if } P_i \geq \alpha \\ 0, & \text{if } |P_i| < \alpha \\ \bar{1}, & \text{if } P_i \leq \bar{\alpha} \end{cases}$
        $B_i = \begin{cases} 1, & \text{if } Q_i \geq \alpha \\ 0, & \text{if } |Q_i| < \alpha \\ \bar{1}, & \text{if } Q_i \leq \bar{\alpha} \end{cases}$
    iii. $S'_i = P_i - r.C_i$, $D'_i = Q_i - r.B_i$
    iv. $S_{i+1} = S'_{i+1} + C_i$, $D_{i+1} = D'_{i+1} + B_i$
    v. Set: $E_{i+1}$ = sign ($S_{i+1}$) and $F_{i+1}$ = sign ($D_{i+1}$)
    vi. Define PE and PF as:
        PE = PE ø $E_{i+1}$, PF = PF ø $F_{i+1}$
        Where K ø L represents sign of K if $K \neq 0$
        and otherwise it represents sign of L.
    vii. M = PE. PF
c. If (PE = 0) or (PF = 0) then |X| = |Y|
    Otherwise, if (M = 1) then |X| > |Y|
        Else |X| < |Y|.
```

Computations at ith iteration are shown in Fig. 5 as the shaded region where the symbols may be interpreted as per algorithmic operations.

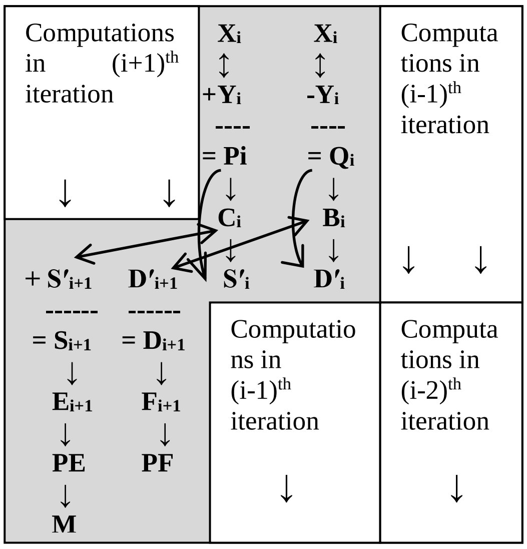

Computations in (i+1)th iteration
X_i
$+Y_i$
$= P_i$
X_i
$-Y_i$
$= Q_i$
Computations in (i-1)th iteration
C_i
B_i
$+ S'_{i+1}$
D'_{i+1}
S'_i
D'_i
$= S_{i+1}$
$= D_{i+1}$
E_{i+1}
F_{i+1}
PE
PF
M
Computations in (i-1)th iteration
Computations in (i-2)th iteration

Fig. 5: Computations in ith iteration

4. RESULTS AND DISCUSSION

For the sake of clarifying its step-by-step operations, at first, the proposed algorithm, algorithm 4, is to be tested on some higher-radix input. For example, consider, two radix-4 ordinary SDNs, X and Y where $X = 12\bar{1}01$, $Y = \bar{1}0\bar{2}\bar{1}\bar{1}$. The iterative computations for MCN of X and Y are presented in table 1.

Table 1: MCN of X and Y as an Example

Iterations (i) → / Temporaries↓	4	3	2	1	0	$\bar{1}$
P_i	0	2	$\bar{3}$	$\bar{1}$	0	0
Q_i	2	2	1	1	2	0
C_i	0	1	$\bar{1}$	0	0	0
B_i	1	1	0	0	1	0
S'_i	0	$\bar{2}$	1	$\bar{1}$	0	0
D'_i	$\bar{2}$	$\bar{2}$	1	1	$\bar{2}$	0
S_{i+1}	0	1	$\bar{3}$	1	$\bar{1}$	0
D_{i+1}	1	$\bar{1}$	$\bar{2}$	1	2	$\bar{2}$
E_{i+1}	0	1	$\bar{1}$	1	$\bar{1}$	0
F_{i+1}	1	$\bar{1}$	$\bar{1}$	1	1	$\bar{1}$
PE	0	1	1	1	1	1
PF	1	1	1	1	1	1
M	0	1	1	1	1	1

Therefore |X| > |Y| which is obviously holds true.

The CAA presented in this paper may be conceptually extended for GSDNS, obviously including BSDNS. However, the addition/ subtraction rules for BSDNS (in broad sense for GSDNS) are known to be fairly complex ([5], [7]). So the direct scaling of the OLMCR presented in this paper may be inapplicable beyond the OSDNS owing to possibly large area, delay and power requirements.

5. CONCLUSION

MCR design is a complex problem for any mode of computing, conventional or unconventional. Even in the existing literature of OLA no report is available on designing MCR. In this paper, a simple and efficient OLMCR for higher-radix is proposed on the basis of interpreting MCN in terms of additions/subtractions and SDTNs [12] as well as circulating the input/ output-digits of both SD-addition/subtraction and SDTN serially from MS-to-LS position. Consequently in OLA platform MCN may also be pipelined with other CA operations, leading to achieve faster processing speed as a whole. The future work of the author would be investigating OLMCR using BSDNS.

REFERENCES

[1] K.S. Trivedi, M.D. Ercegovac, "On-Line Algorithms for Division and Multiplication", IEEE Transactions on Computers, vol. C-26, no.7, 1977, pp. 681-687.

[2] M.D. Ercegovac, "On-Line Arithmetic: An Overview", in Proceedings of Proc.SPIE0495, Real-Time Signal Processing VII, 1984, pp. 86-93.

[3] M.D. Ercegovac, "On Left-to-Right Arithmetic", in Proceedings of 51st Asilomar Conference on Signals, Systems and Computers, 2017, pp. 750-754.

[4] A. Avizienis, "Signed Digit Number Representations for Fast Parallel Arithmetic," IRE Transactions on Electronic Computers, vol. EC-10, no. 3, 1961, pp. 389-400.

[5] I. Koren, Computer Arithmetic Algorithms, 2nd Edition, A.K. Peters Ltd., Natick, MA, 2002.

[6] M.D. Ercegovac, T. Lang, Digital Arithmetic, Morgan Kaufmann Publishers, 2004.

[7] B. Parhami, "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations", IEEE Transactions on Computers, vol. 39, 1990, pp. 89-98.

[8] Y. Zhao, J. Wickerson, G. A. Constantinides, "An Efficient Implementation of Online Arithmetic", in Proceedings of Intl. Conf. on Field-Programmable Technology (FPT), 2016, pp. 69-76.

[9] A. Tisserand, P. Marchal, C. Piguet, "An On-Line Arithmetic Based FPGA for Low-Power Custom Computing", in Proceedings of Workshop on Field Programmable Logic and Applications, UK, 1999, pp.264-27.

[10] S. Veeramachaneni, M.K. Krishna, L. Avinash, R.P. Sreekanth, M. B. Srinivas, "Efficient Design of 32-bit Comparator using Carry Look-Ahead Logic", in Proceedings of IEEE Northeast Workshop on Circuits and Systems (NEWCAS), 2007, pp. 867-870.

[11] S. Kumar, C-H. Chang, T.F. Tay, "New Algorithm for Signed Integer Comparison in $\{2^{n+k}, 2^n-1, 2^n+1, 2^{n\pm1}-1\}$ and Its Efficient Hardware Implementation", IEEE Transactions on Circuits and Systems, vol. 64, issue 6, 2016, pp. 1481-1493.

[12] M. S. Chakraborty, A. C. Mondal, S. K. Sao, "Towards Relating Some Methods of Signed-Digit Arithmetic", Chapter 7, Advances in Mathematical Sciences, ISBN 978-93-8643-44-6, Edited by Department of Mathematics, St. Thomas College, Thrissur: India, 2018, pp. 47-54.

[13] T. Lynch, M.J. Schulte, "A High-Radix On-Line Arithmetic for Credible and Accurate Computing", The Journal of Universal Computer Science, vol. 1, no. 7, 1995, pp. 439-453.

[14] R. Hartley and P. Corbett, "Digit-Serial Processing Techniques. IEEE Transactions on Circuits and Systems", vol. 37, no. 6, 1990, pp.707-719.

[15] M. J. Irwin and R. M. Owens, "Digit-Pipelined Arithmetic as Illustrated by the Paste-Up System: A Tutorial", Computer, vol. 20, no. 4, 1987, 61-73.

[16] K. Shi, D. Boland, and G. A. Constantinides, "Efficient FPGA Implementation of Digit Parallel Online Arithmetic Operators", in Proceedings of Intl. Conference on FPT, China, 2014.

[17] K. Shi, D. Boland, E. Stott, S. Bayliss and G. A. Constantinides, "Datapath Synthesis for Overclocking: Online Arithmetic for Latency Accuracy Trade-Offs", in Proceedings DAC, USA, 2014.

[18] W. G. Natter and B. Nowrouzian, "Digit-serial Online Arithmetic for High-Speed Digital Signal Processing Applications", in Proceedings of 35th Asilomar Conf. On Signals, Systems and Computers, 2001, pp. 171-176.

[19] M. Dimmler, A. Tisserand, U. Holmbeg, and R. Longchamp. "On-Line Arithmetic for Real-Time Control of Microsystems", IEEE/ASME Transactions on Mechatronics,, vol. 4, no. 2, 1999, pp. 213-217.

[20] M. S. Chakraborty, "Reverse Conversion Schemes for Signed-Digit Number Systems: A Survey", Journal of Institution of Engineers (I): Series B, Vol. 97, 2016, pp. 589-593.

[21] K. G. Smitha, A. H. Fahmy, A. P. Vinod, "Redundant Adders Consume Less Energy", in Proceedings of IEEE APC on Circuits and Systems, Singapore, 2006, pp. 422-425.

[22] D. Crookes, M. Jiang, "Using Signed Digit Arithmetic for Low Power Multiplication", Electronics Letters, 2007, pp. 13-14.

[23] D. S. Phatak, S. Kahle, H. Kim, J. Lue, "Hybrid Signed Digit Representation for Low Power Arithmetic Circuits", in Proceedings of Low Power Workshop in Conjunction with ISCA, Barcelona: Spain, 1998.

[24] H. R. Srinivas, K. K. Parhi, "A Radix 2 Shared Division/Square Root Algorithm and Its VLSI Architecture", Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, vol. 21, 1999, pp. 37-60.

[25] G. C. Cardarilli, S. Pontarelli, M. Re, A. Salsano, "Fault Tolerant Design of Signed-Digit based FIR Filters", in Proceedings of IEEE International Symposium on Circuits and Systems, Greece, 2006, pp. 2809-2812.

[26] S. Venkatachalapathy, Signed Digit Representation of Numbers, M. S. Dissertation, Oregon State University: USA, 2006.

[27] A. Kaivani, S. Ko, "Floating Point Butterfly Architecture Based on Binary Signed Digit Representation", IEEE Transactions on Very Large Scale Integration, vol. 24, 2016, pp. 1208-1211.

[28] M. S. Chakraborty, A Study of Reverse Conversion Algorithms for Signed-Digit Number Systems, Ph. D. Thesis, The University of Burdwan: India, 2019.

[29] T. Srikanthan, S. K. Lam, M. Suman, "Area-Time Efficient Sign-Detection Technique for Binary Signed-Digit Number System", IEEE Transactions on Computers, Vol. 53, 2004, pp. 69-72.

AUTHOR'S PROFILE

Dr. Madhu Sudan Chakraborty holds Master of Computer Applications degree (N.I.T, Jamshedpur, India) and Ph. D. (Burdwan University, India). He has been serving as an Asst. Professor (Sr. Level) of Computer Science at Indas Mahavidyalya, Bankura, WB, India-722205, since 2007. He is a member of IEEE and IAENG. His thrust area is theoretical computer science, including computer arithmetic. He has to his credit several publications in reputed national/international journals, conference proceedings and book chapters, including those belong to IEEE and Springer.