Advances and Applications in Mathematical Sciences
Volume 21, Issue 6, April 2022, Pages 3479-3491

ON OVERFLOW HANDING IN BINARY SIGNED-DIGIT ARITHMETIC

MADHU SUDAN CHAKRABORTY

Department of Computer Science
Indas Mahavidyalaya, Indas
West Bengal, 722205, India
E-mail: mailmschakraborty@rediffmail.com

Abstract

Signed-digit number systems are growing with several attractive features, providing underpinning for online arithmetic too. However, signed-digit numbers appear to be more susceptible to overflow due to carrying intrinsic redundancy. Obviously the practical applications of signed-digit arithmetic may have been constrained by overflow as one of its performance bottlenecks and so efficient handling of overflow is a prerequisite for the survival of signed-digit arithmetic. In the traditional signed-digit arithmetic the known method for overflow handling is fairly complex whereas in online arithmetic no method for overflow handling has been formulated yet. In this paper, one of the most attractive classes of signed-digit number systems, binary signed-digit number system, is further investigated for overflow handling. The investigations result in proposing two different algorithms. The first algorithm is applicable to the traditional signed-digit arithmetic domain, showing more prospects in wider context. The second algorithm may run in online mode.

1. Introduction

Adders serve as one of the basic building blocks of integrated arithmetic units (AUs) and so incorporating faster adders is a prerequisite for designing faster processors [1]. However, for the conventional radix-complement number system addition necessitates carry propagation which may cause drastic reduction in the processor's speed. For remediation some alternative computational philosophies have been aired with unconventional number systems (UCNSs) [1], allowing carry-free or limited-carry

2020 Mathematics Subject Classification: 68W10, 68W27, 97P20.
Keywords: Signed-Digit Number Systems, Computer Arithmetic Algorithms, Online Arithmetic, Binary Signed-Digit, Overflow Handling.
Received July 7, 2021; Accepted August 28, 2021

addition/subtraction. Signed-digit number system (SDNS) is an UCNS ([1]-[2]), which can restrict carry/borrow chains during addition/subtraction to some closed neighbourhood for all counterpart operand positions. In other words, SDNS offers faster addition/subtraction, employing regular arithmetic circuits, which, in turn, catalyzes faster execution of some more complex arithmetic operations (AOs), including multiplication and division [1]. In addition for the last few decades several arithmetic algorithms (AAs) have been developed in signed-digit (SD) environment to satisfactorily support cryptography ([3]-[4]) low-power consuming AOs ([5]-[6]), fault-tolerance [7] and on-line arithmetic (OLA) ([8]-[9]) too. However, one basic problem with signed-digit arithmetic (SDA) is that SDNS intrinsically carries redundancy which makes it more susceptible to overflow [1]. This is what strives to restrict the practical applications of SDNS as one of its major performance bottlenecks, besides problems like reverse conversion (RC) ([1], [10]) and magnitude comparison [1]. Clearly efficient handling of overflow is a prerequisite for the survival of the SDNS. In the traditional SDA the known method for overflow handling (OH) is fairly complex [11] whereas in OLA no method for OH has been formulated yet. In this paper one of the most attractive classes of SDNSs, binary signed-digit number system (BSDNS), will be further investigated for OH with greater efficiency. The remaining portion of the article will be organized with five sections as follows: Some stakeholder-issues and their backgrounds will be discussed in brief in section 2. The existing AA for OH in SDA environment will be revisited in section 3. Two new AAs will be proposed for OH in BSDNS platform in section 4. One algorithm will be the traditional SDA-compliant whereas the other will be OLA-compatible. The comparative merit of the first algorithm over its potential contenders will be presented in section 5 in the wider context. Finally the presentation will be ended in section 6 with concluding remarks and future directions.

2. The Background

In this section, for accordant presentation the mathematical formulations of SDNS as well as some other stakeholder-issues for OH are revisited in brief.

2.1. SDNS. In general the radix-r $(r \geq 2)$ SDNS is defined as a positional

number system over the DS $S = \{\overline{\alpha}, \overline{\alpha}+1, \ldots, \overline{1}, 0, 1, \ldots, \beta-1, \beta\}$ where $\alpha \geq 0, \beta \geq 0$ and $\alpha + \beta + 1 > r$. It means every number X, expressed in radix-r signed-digit form as: $X_{n-1}X_{n-2}\ldots X_0 \; \forall X_i \in S$, must observe: $X = \sum_{i=0}^{n-1} X_i$. Different classes of SDNS may appear with varying α and β, which are termed as SDNSs as a whole. In simplest form $\alpha = \beta = 1$ and it results $S = \{\overline{1}, 0, 1\}$ which is the BSDNS.

2.2. SD-Encodings. As the digital devices are driven by binary arithmetic principles at an extreme arithmetic level, some mapping from the signed digit-set (DS) to the traditional binary DS is mandatory. For the BSDNS two popular encodings are: two's-complement encoding (TCE) and positive-negative encoding (PNE) as specified in Table 1. In this study, TCE will be employed throughout for BSD encoding.

Table 1. Encodings of binary signed-digits (BSDs).

Binary Signed-Digit (BSD)	Binary Encoding	
	PNE	TCE
$\overline{1}$	01	11
0	00 (or 11)	00
1	10	01

2.3. Redundancy in SDNS. In SD-platform some numbers may have two or even more representations and so SDNS is also called the redundant number system. For SDNSs intrinsic redundancy is essential, because carry-free/limited-carry addition/subtraction is not possible, otherwise. For instance in BSDNS-platform $(-7)_{10}$ can be represented as $(0\overline{1}\,\overline{1}\,\overline{1})_2$, $(\overline{1}001)_2$ and $(\overline{1}1001)_2$. Then if the computing device can support 4-digit number representation only (at least hypothetically) the representation $(\overline{1}1001)_2$ will cause an overflow. In this case the overflow is resolvable by using some alternative representation for $(-7)_{10}$ say $(0\overline{1}\,\overline{1}\,\overline{1})_2$ or $(\overline{1}001)_2$ as presented in this example. This type of overflow is termed as apparent overflow. Obviously the apparent overflow is manageable for not causing computational

errors. The other type of overflow definitely causes computational errors and it is known as actual overflow.

2.4. Sign-Detection. Sign of the most-significant non-zero digit uniquely determines the sign of a signed-digit number (SDN). For a given binary-SDN $Z = Z_{n-1}Z_{n-2}\ldots Z_1Z_0$ every sign-detection (SDTN) algorithm must conform the rules presented in Table 2.

Table 2. Rules for Sign-Detection.

X_i	S_i	S_{i+1}
$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	$\bar{1}$
$\bar{1}$	1	$\bar{1}$
0	$\bar{1}$	$\bar{1}$
0	0	0
0	1	1
1	$\bar{1}$	1
1	0	1
1	1	1

Here negative, zero and positive signs are denoted by $\bar{1}, 0$ and 1 respectively. Initially $S_0 = 0$ and finally S_n gives the required sign. The methodological alliance between sign-detection and OH will be elaborated in details in the next session. In this study, apart from BSDs, for representing sign-information TCE will be used.

2.5. Online Arithmetic. In OLA for every operation the concerned phases are executed serially and only in the direction of most-significant-digit (MSD) to least-significant-digit (LSD). In addition some phases of two different operations may coincide over the same time slot and consequently as a whole faster execution of a group of operations may be possible [8]. OLA is a recent trend in unconventional computing and it is an exclusive characteristic

of SDNSs to admit OLA. In the recent years online arithmetic algorithms (OLAAs) for complex AOs have been being investigated too ([9], [12]).

3. Revisiting the Existing OH-algorithms

Suppose that a n-digit BSD-number (BSDN) $X = X_{n-1}X_{n-2}\ldots X_0$ is to be tested for overflow. It is needless to say that any question of further investigating an overflow may arise only when $X_n \neq 0$. The existing approach of OH is to employ two independent sub-algorithms, one for sign-detection and the other for overflow-resolution [11]. On the basis of the rules presented in Table 2 an AA for SDTN of BSDNS is presented as Algorithm 1.

Algorithm 1. Sign-Detection

1. Initialize: $S_{0,1} = 0$, $S_{0,0} = 0$

2. For $i = 0$ to $n-1$ do:

a. Compute: $S_{i+1,1} = X_{i,1} + \overline{X_{i,0}} \cdot S_{i,1}$, $S_{i+1,0} = X_{i,0} + S_{i,0}$

When X_n and S_n contradict in sign, it is an apparent overflow. Otherwise, it would be a real overflow. When the storage unit is limited to n-bit, in case of real overflow the desired SD-output can never be restored equivalently. However for an apparent overflow the desired SD-output can be restored equivalently [11]. In this regard all cases are shown in Table 3 where $T = 0$ and $T = 1$ represent actual and apparent overflow respectively.

Table 3. Overflow Type Checking.

X_n	S_n	T
$\bar{1}$	$\bar{1}$	0
$\bar{1}$	0	0
$\bar{1}$	1	1
0	$\bar{1}$	N/A
0	0	N/A

0	1	N/A
1	$\bar{1}$	1
1	0	0
1	1	0

The existing approach for apparent-overflow resolution [11] uses a BSD-variable (B_i) as the driver which works as follows: Initialize: $B_{n-1} = X_n$ and $i = n-1$. Then compute and move towards the lower-significant-digit positions on par with Table 4.

Table 4. Existing Rules for Overflow Resolution [11].

X_i	B_i	Y_i	B_{i-1}
$\bar{1}$	$\bar{1}$	N/A	N/A
$\bar{1}$	0	$\bar{1}$	0
$\bar{1}$	1	1	0
0	$\bar{1}$	$\bar{1}$	$\bar{1}$
0	0	0	0
0	1	1	1
1	$\bar{1}$	$\bar{1}$	0
1	0	1	0
1	1	N/A	N/A

During overflow resolution the cases marked by N/A do not occur. B_i becomes 0 either when $(X_{i+1} = 1$ and $B_{i-1} = \bar{1})$ or when $(X_{i+1} = \bar{1}$ and $B_{i-1} = 1)$. Once B_i becomes $0, Y_i = X_i \ \forall \ i \geq 0$ i.e. no more I/O transformation is required. On the basis of generating characteristic equations for Table 3 and Table 4 the traditional overflow resolution process is represented as Algorithm 2 as follows:

Algorithm 2. Overflow Resolution

1. Compute: $T = X_{n,1} \cdot S_{n,0} + X_{n,0} \cdot S_{n,1}$

2. If $(T = 0)$ Output: "Actual Overflow"

3. Otherwise

a. Initialize: $B_{n-1,0} = X_{n,0}$, $B_{n-1,1} = X_{n,1}$

b. For $i = n - 1$ to 1 do as follows:

i. Compute: $Y_{i,1} = X_{i,1}\overline{B_{i,0}} + B_{i,1}$, $Y_{i,0} = X_{i,0}\overline{B_{i,1}} + B_{i,0}$

ii. Compute: $B_{i-1,1} = \overline{X_{i,0}} \cdot B_{i,1}$, $B_{i-1,0} = \overline{X_{i,1}} \cdot B_{i,1}$

c. Compute: $Y_{0,1} = X_{0,1} \cdot \overline{B_{0,0}} + B_{0,1}$, $Y_{0,0} = X_{0,0} \cdot \overline{B_{0,1}} + B_{0,0}$

d. Output "It is an apparent overflow and resolved as:", Y

4. New Proposals for OH

In this section two new algorithms are proposed for OH. The first algorithm is applicable to the traditional binary-SDA domain whereas the second algorithm may run as an OLAA.

4.1. OH in Traditional Binary-SDA Domain: Proposing a Novel Algorithm

The proposed algorithm exploits Lemma 1 which is as follows:

Lemma 1. *For every BSDN* $Z(\neq 0)$, $\exists$ *some BSDN Y such that every digit of Y is non-zero, excluding the sequence of zero*(*s*) *covering the least-significant position of Y, if any.*

As $(0\ldots01)_2 = (1\ldots\overline{1}\,\overline{1})_2$ and $(0\ldots0\overline{1})_2 = (\overline{1}\ldots11)_2$ hold true, Lemma 1 immediately holds true. Let the typical representation of Y is called Intermediate Zero-Free Binary Signed-Digit Representation (IZFBSDR). The rules for converting ordinary BSD-representation into the equivalent IZFBSDR are listed in Table 5.

Table 5. Proposed Rules for IZFBDSR.

X_i	X_{i-1}	Y_i	Y_{i-1}
$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$
$\overline{1}$	0	$\overline{1}$	0
$\overline{1}$	1	$\overline{1}$	1
0	$\overline{1}$	$\overline{1}$	1
0	0	0	0
0	1	1	$\overline{1}$
1	$\overline{1}$	1	$\overline{1}$
1	0	1	0
1	1	1	1

On realizing the governing Boolean expressions for Table 4, the conversion algorithm may be presented as Algorithm 3.

Algorithm 3. OH-proposal for the traditional SDA-platform

1. Set:

   a. Compute: $Y_{1,1} = X_{1,1} + \overline{X_{1,0}} \cdot X_{0,1}$, $Y_{1,0} = X_{1,0} + \overline{X_{i,1}} \cdot X_{0,0}$

   b. Compute: $Z_{0,1} = X_{1,0} \cdot X_{0,1} + \overline{X_{1,0}} \cdot \overline{X_{0,1}} \cdot X_{0,0}$

   $$Z_{0,0} = X_{0,0}$$

2. For $i = 2$ to $n-1$ do as follows:

   a. Compute: $Y_{i,1} = X_{i,1} + \overline{X_{i,0}} \cdot Y_{i-1,1}$, $Y_{i,0} = X_{i,0} + \overline{X_{i,1}} \cdot Y_{i-1,0}$

   b. Compute: $Z_{i-1,1} = X_{i,0} \cdot Y_{i-1,1} + \overline{X_{i,0}} \cdot \overline{Y_{i-1,1}} \cdot Y_{i-1,0}$

   $$Z_{i-1,0} = Y_{i-1,0}$$

3. Compute: $T = X_{n,1} \cdot Y_{n-1,0} + X_{n,0} \cdot Y_{n-1,1}$

4. If $(T = 0)$ Output: "Actual Overflow"

5. Else

a. Rectify: $Y_{n-1,1} = X_{n,1}$, $Y_{n-1,0} = X_{n,0}$

b. Output "It is an apparent overflow and resolved as:", Y

4.2. OH in Online Arithmetic Domain: Proposing an Algorithm

The proposed algorithm, Algorithm 4, is an extension of Algorithm 2. The Algorithm 4 uses a binary variable, D_i, to assess the overflow type (actual or apparent) for i^{th} iteration $\forall\ i \in [0, n-1]$ where $D_i = 0$ means the overflow is actual and $D_i = 1$ means the overflow status is not clear yet. Initially $D_{n-1} = 1$. D_i becomes 0 either when $(X_{i+1} = 1$ and $B_{i+1} = 1)$ or when $(X_{i+1} = \bar{1}$ and $B_{i+1} = \bar{1})$. Once D_i becomes 0, no more computation is required, except propagating $D_i = 0$ to become $D_i = 0$. On the other hand for $D_i = 1$ the overflow resolution process is obviously continued. Finally in case of occurrence of $D_i = 1$, it may be concluded that the overflow has been resolved with an equivalent substitute. The driving rules for Algorithm 4 are presented in Table 6.

Table 6. Rules for the Proposed OLAA for OH.

X_i	B_i	D_i	Y_i	B_{i-1}	D_{i-1}
$\bar{1}$	$\bar{1}$	1	NOP		0
$\bar{1}$	$\bar{1}$	0	NOP		0
$\bar{1}$	1	1	1	0	1
$\bar{1}$	1	0	NOP		0
0	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	1
0	$\bar{1}$	0	NOP		0
0	0	1	0	0	1
0	0	0	NOP		0

0	1	1	1	1	1
0	1	0	NOP		0
1	$\bar{1}$	1	$\bar{1}$	0	1
1	$\bar{1}$	0	NOP		0
1	0	1	1	0	1
1	0	0	NOP		0
1	1	1	NOP		1
1	1	0	NOP		0
1	0	1	$\bar{1}$	0	1
1	0	0	NOP		0

On the basis of generating characteristic equations for Table 6 the OH process may be expressed as Algorithm 4 as follows:

Algorithm 4. OH-proposal for the OLA-platform

1. Initialize: $B_{n-1} = X_n$, $D_{n-1} = 1AA$

2. For $i = n - 1$ to 0 do as follows:

   a. If $D_i = 1$ do as follows:

      i. Compute: $Y_{i,1} = X_{i,1} \cdot \overline{B_{i,0}} + B_{i,1}$, $Y_{i,0} = X_{i,0} \cdot \overline{B_{i,1}} + B_{i,0}$

      ii. Compute: $B_{i-1,1} = \overline{X_{i,0}} \cdot B_{i,1}$, $B_{i-1,0} = \overline{X_{i,0}} \cdot B_{i,0}$

      iii. Compute: $D_{i-1} = \overline{X_{i,1}} \cdot \overline{X_{i,0}} + \overline{X_{i,1}} \cdot \overline{B_{i,0}} + \overline{X_{i,0}} \cdot \overline{B_{i,1}}$

   b. Else do

      i. NOP

      ii. set: $D_0 = 0$

3. If $(D_0 = 1)$ Output "It is an apparent overflow and resolved as:", Y

4. Else Output: "It is an actual overflow"

5. Results and Discussion

In this paper two new AAs for OH in BSDNS have been proposed. One AA, Algorithm 3, is traditional BSDNS-compliant. The other AA, Algorithm 4, is OLA-compatible. Algorithm 3 seems to air a novel philosophy in the literature of SDA. In order simplify the analysis of the comparative merit of Algorithm 3, let its step 1 through step 2 are marked as a whole as the pre-processing stage and its rest steps are collectively marked as the resolution stage. It is true that the area, delay and power overheads of the pre-processing stage of Algorithm 3 may be significant, rather than being negligible and experimental study is needed to accurately quantify those parameters. However, still the Algorithm 3 is worthy for further consideration regardless of its immediate comparative merit or demerit over its existing counterpart [11], as once the execution of the pre-processing stage is completed, it may provide constant-time solution not only to the OH problem, but also to the RC-problem. It is well known that the RC for SDNSs necessarily involves high overheads and no circuit faster than log-depth time complexity is known [10]. However, as the pre-processing stage of Algorithm 3 necessarily yields IZFBSDR and in IZFBSDR the conversion control information [10] for every digit position is directly implied by the sign of its immediate lower-significant digit, constant-time RC may become possible too. Further the integration of AUs for RC and OH-and even radix-complement to sign-magnitude transformation [13] may become possible at ease, with a promise for the higher-degree performance as a whole.

On the other hand, the contribution of the Algorithm 4 to the literature is clearly evident from the fact that it attempts to present the first ever proposal for OH over OLA-platform.

However, some limitations of this study are that it is purely theoretical and for investigating the OH-problem and its prospective solutions; only linear algorithmic models are considered in line with some latest reports published in the allied fields [14]. However, for stand-alone SDTN, the use of non-linear network, particularly optimized reverse tree network [15], seems to be the better option. In other words, Algorithm 1 and Algorithm 3 may be

outperformed by the corresponding non-linear models. Obviously it would be interesting for carrying the research forward in this direction.

6. Conclusion

Efficiency in OH is very important for keeping the future prospects of SDNSs alive. In this paper, one of the most attractive classes of SDNDs, BSDNS, is considered and two algorithms for OH are proposed.

The first algorithm, Algorithm 3, works on the traditional SDA domain whereas the second algorithm, Algorithm 4, may run in the OLA domain. It is true that the pre-processing stage of Algorithm 3 may attract additional area, delay and power as overheads. However, still Algorithm 3 is worthy for further consideration as the immediate outcome of its pre-processing stage is IZFBSDR and IZFBSDR may serve as the basis for constant-time, one-stop solution for overflow resolution as well as some other complex problems of SDA, including RC. On the other hand, the second algorithm, Algorithm 4, appears to be the first ever proposal for OH in OLA mode to the author's best knowledge and belief. As a subject matter of future studies by the author, non-linear variants of some stakeholder algorithms of this study, namely SDTN and IZFBSDR, would be considered, for possible enhancement in OH-performance. The future study would be theoretical as well as experimental.

References

[1] I. Koren, Computer arithmetic algorithms, 2nd edn, CRC Press, London, (2001).

[2] A. Avizienis, Signed-digit number representations for fast parallel arithmetic, IRE Trans. Electron. Comput. EC-10 (1961), 389-400.

[3] Y. A. Shah, K. Javeed, S. Azmat and X. Wang, Redundant-signed-digit-based high speed elliptic curve cryptographic processor, Journal of Circuits, Systems and Computers 28 (2019), 120-126.

[4] Y. Wang, D. L. Maskell, J. Leiwo and T. Srikanthan, Unified signed-digit number adder for RSA and ECC public-key cryptosystems, In Proceedings of IEEE Asia Pacific Conference on Circuits and Systems, Singapore (2006), 1655-1658.

[5] D. Crookes and M. Jiang, Using signed digit arithmetic for low power multiplication, Electron. Lett. 43 (2007), 613-614.

[6] K. G. Smitha, A. H. Fahmy and A. P. Vinod, Redundant adders consume less energy, In Proceedings of IEEE Asia Pacific Conf. on Circuits and Systems, Singapore (2006), 422-425.

[7] A. Armand and S. Timarchi, Efficient error detection and correction method for 1-out-of-3 binary signed digit adders, International Journal of Electronics 106 (2019), 1427-1440.

[8] M. D. Ercegovac and T. Lang, Digital arithmetic, morgan kaufmann publishers (An Imprint of Elsevier), San Francisco, (2004).

[9] M. S. Chakraborty, Designing an on-line magnitude comparator for higher-radix, International Journal of Information Technology and Electrical Engineering 9 (2020), 92-98.

[10] M. S. Chakraborty, Reverse conversion schemes for signed-digit number systems: a survey, Journal of Institution of Engineers (I): Series B 97 (2016), 589-593.

[11] B. Parhami, Zero, Sign, and overflow detection schemes for generalized signed-digit arithmetic, In Proceedings of Twenty-Second Asilomar Conference on Signals, Systems and Computers (1988), 636-639.

[12] W. E. Ferguson, J. Bingham, L. Erkök, J. R. Harrison and J. Leslie-Hurd, Digit serial methods with applications to division and square root, In IEEE Transactions on Computers 67 (2018), 449-456.

[13] M. S. Chakraborty and A. C. Mondal, Reverse conversion of signed-digit number systems: transforming radix-complement output, Indonesian Journal of Electrical Engineering and Computer Science 4 (2016), 665-669.

[14] R. K. Barik, M. Pradhan and R. Panda, Efficient conversion technique from redundant binary to non-redundant binary representation, Journal of Circuits, Systems and Computers 26 (2017), 1750135.

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