Notes on "A Novel Conversion Scheme from a Redundant Binary Number to Two's Complement Binary Number for Parallel Architectures" proposed by Choo et.al.

M S Chakraborty
Asst Prof in Comp Sc
Indas Mahavidyalaya
Bankura 722205

T Ghosh
Asst Prof in Comp Sc
Sammilani College
Bankura 722102

A C Mondal
Asso Prof in Comp Sc
Burdwan University
Burdwan 713104

ABSTRACT
In this article, it is shown that although the reverse conversion scheme for binary signed-digit number system proposed by Choo et. al. can not support full parallelism; the rules on which it is based are correct. In this connection, a mathematical induction technique is used to validate the decomposition rules. Accordingly, it can be inferred that the reverse conversion scheme for binary signed-digit number system proposed by Veeramachaneni et. al. works correctly and performs reverse conversion in Ω (log n) time, where, n is the input size. As a consequence, the scheme by Veeramachaneni et. al. need to be considered as a potential contender of the more recent schemes for the same.

General Terms
Unconventional Number System, Computer Arithmetic Algorithm.

Keywords
Reverse conversion scheme, binary signed-digit number system, parallelism, validation, comparative study.

1. INTRODUCTION
Signed-digit number system [1][2] is still an active area of study [3][4]. Reverse conversion (RC) of a signed-digit number system means converting signed-digit numbers of the system back into the conventional form, such as, sign-magnitude representation, radix-complement representation. RC, particularly, RC of binary signed-digit number (BSDN) system is an important research problem [5][6] because, the accustomed bus architectures for the digital signal processing and also operations of standard peripheral devices are still based on the two's complement representation. But, yet, the available algorithms for the same involve significant overheads in form of time, area and power.

In the literature of RC for BSDN system, Choo et. al. [7] claimed a breakthrough by proposing a so-called fully-parallel RC scheme (RCS) for BSDN system. In [7], the basic concept is to decompose given n-digit BSDN input, F, into two separate BSDNs (components), X and Y, along with the status output, SO, in such a way that X and SO●Y (i.e. SO is prefixed to Y) can be added without any carry propagation chain to produce the output in correct two's-complement form. In this connection, typical rules were proposed for decomposition and addition. Suppose that $F = f_{n-1}f_{n-2}\ldots f_0$, $X = x_{n-1}x_{n-2}\ldots x_0$, $Y = y_{n-1}y_{n-2}\ldots y_0$, where, f_0, x_0 any y_0 represent the LSD of F, X and Y respectively.

The decomposition rules state that:

$x_i = 1, \forall\, i \in [0, n-1]$ (1)

SO = 0, if $f_{n-1} = 1$; otherwise, SO = $\bar{1}$. (2)

Set, y_n = SO

$y_0 = 1$, if $f_0 = 0$; otherwise, $y_0 = 0$. (3)

Rules for computing $y_i, \forall\, i \in [1, n-1]$ are given by table 1.

Table 1. Computing $y_i, \forall\, i \in [1, n-1]$ [7]

Inputs		Output
f_i	f_{i-1}	(y_i)
$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	$\bar{1}$
$\bar{1}$	1	0
0	$\bar{1}$	0
0	0	0
0	1	1
1	$\bar{1}$	$\bar{1}$
1	0	$\bar{1}$
1	1	0

The projected addition rules for summing up the decomposed components are given by table 2, where, D/C means DON'T CARE condition. The rules presented in table 2 are a subset of the rules originally presented in [7], subjected to the condition that $x_i = 1$, for all $x_i \in [0, n-1]$. In table 2, s_i is the intermediate sum digit at to $(i+1)^{th}$ position and c_i the intermediate carry-out from the same position. Obviously, c_i acts as the status signal for the computing intermediate outputs corresponding to the next higher significant position.

Table 2. Projected Addition Rules [7]

Type	x_i	y_i	c_{i-1}	c_i	s_i
1	1	0	1	1	$\bar{1}$
			$\bar{1}$, 0	0	1
2	1	1	D/C	1	0
3	1	$\bar{1}$	D/C	0	0

Although, Choo et. al. [7] claimed to achieve full parallelism, no proof for the correctness of its rules and their interoperability were given in the original paper. The claim for achieving full parallelism in [7] was invalidated by the computer arithmetic community [5][6][8] straightly due to non-compliance with [9]. But, none of the authors pointed out the root cause of the incorrectness of [7]: whether some rule(s) is incorrect or their inter-operability fails to ensure full parallelism. Although, in [10] the decomposition rules of [7] exploited with a parallel prefix network and the outcome of the study was found to be significant; again, no proof for correctness of the underlying rules was given.

The remainder of this paper is organized as follows: The correctness of the decomposition rules and addition rules will be verified followed by an example to invalidate the claim for full parallelism in [7]. Then, scheme [10] will be reviewed as a derivative of [7].

2. DECOMPOSITION RULES [7]: ON THE CORRECTNESS

Since Y = F – X and X is a string of 1s', y_i should be computed in such a way $c_i \bullet y_i = f_i\text{-}1+c_{i-1}$, where, SO = c_{n-1}. Let, SO be denoted as SO_{n-1} onwards for simplicity. The correctness of decomposition rules will be proved next by means of mathematical induction.

2.1 Basic of Induction

Suppose that n = 1. The decompositions of f_0 for its different values using rule (2) and rule (3) are shown in table 3:

Table 3. Decomposition of f_0

Input (f_0)	Outputs	
	y_0	SO_0
$\bar{1}$	0	$\bar{1}$
0	1	$\bar{1}$
1	0	0

Since for all inputs listed in table 3, $f_0 = SO \bullet y_0+1$, it is concluded that for n = 1 the decomposition is correct.

2.2 Inductive Step

Suppose that the decomposition rules can work correctly for n = k. It means y_k and SO_k can be computed using f_k and f_{k-1} as shown in table 4.

Table 4. Computing y_k and SO_k using f_k and f_{k-1}

Inputs		Outputs	
f_k	f_{k-1}	y_k	SO_k
$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	$\bar{1}$	$\bar{1}$
$\bar{1}$	1	0	$\bar{1}$
0	$\bar{1}$	0	$\bar{1}$
0	0	0	$\bar{1}$
0	1	1	$\bar{1}$
1	$\bar{1}$	$\bar{1}$	0
1	0	$\bar{1}$	0
1	1	0	0

The relationship shown in table 4 can be extended for n = k + 1 considering three consecutive BSDs f_{k+1}, f_k, f_{k-1} as presented in table 5, where, $SO_{k+1} \bullet y_{k+1} = f_{k+1}+SO_k$ (corresponding to the same values of $f_k f_{k-1}$ as in table 4) – 1.

Table 5. Computing y_{k+1} and SO_{k+1} using f_{k+1}, f_k and f_{k-1}

Inputs			Outputs	
f_{k+1}	f_k	f_{k-1}	y_{k+1}	SO_{k+1}
$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	$\bar{1}$	0	$\bar{1}$	$\bar{1}$
$\bar{1}$	$\bar{1}$	1	$\bar{1}$	$\bar{1}$
0	$\bar{1}$	$\bar{1}$	0	$\bar{1}$
0	$\bar{1}$	0	0	$\bar{1}$
0	$\bar{1}$	1	0	$\bar{1}$
1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0
1	$\bar{1}$	0	$\bar{1}$	0
1	$\bar{1}$	1	$\bar{1}$	0
$\bar{1}$	0	$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	0	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	1	$\bar{1}$	$\bar{1}$
0	0	$\bar{1}$	0	$\bar{1}$
0	0	0	0	$\bar{1}$
0	0	1	0	$\bar{1}$
1	0	$\bar{1}$	$\bar{1}$	0
1	0	0	$\bar{1}$	0
1	0	1	$\bar{1}$	0
$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$
$\bar{1}$	1	0	0	$\bar{1}$
$\bar{1}$	1	1	0	$\bar{1}$
0	1	$\bar{1}$	$\bar{1}$	0
0	1	0	$\bar{1}$	0
0	1	1	$\bar{1}$	0
1	1	$\bar{1}$	0	0
1	1	0	0	0
1	1	1	0	0

In table 5 it is found that the decomposition is independent of LSD for any group of three BSDs. Then, on eliminating the redundancy table 5 becomes resemble to table 4 with k = k + 1. Therefore, it can be concluded that the decomposition rules work correctly for all input sizes.

3. ADDITION RULES [7]: ON THE CORRECTNESS

Addition rules are immediately found to be correct.

4. FULL PARALLELISM [7]: ON THE CLAIM

Type 1 addition rule as listed in table 2 indicates that in some situation (where $x_i = 1$ and $y_i = 0$) the intermediate sum-digit at $(i+1)^{th}$ position as well as the carry-out from the same position depend on the intermediate carry-out from i^{th} position. It may generate a chain of dependencies causing the carry to propagate. The point is elaborated with example 1, where, $F = 100\bar{1}$ as presented in Fig. 1. Example 1 shows that situation may occur where the type 1 additions are to be performed repeatedly at adjacent positions. Yet, full-parallelism can be achieved if it is ensured that in the underlying situation only one specific carry value (0 or 1 or $\bar{1}$) always propagates regardless of the overall input pattern. But, this criterion also does not hold. Suppose that, $F = 000\bar{1}0000$.

Here, $c_0 = 1$ and the carry with the same value is propagated up to the 5^{th} position (i.e. i = 4). However, then, $c_4 = 0$ and the carry with the same value is propagated up to the 8^{th} position (i = 7). It can be concluded that the propagation delay for the reverse conversion of a BSDN input may even be proportional to its length. Accordingly, [6] cannot achieve full parallelism.

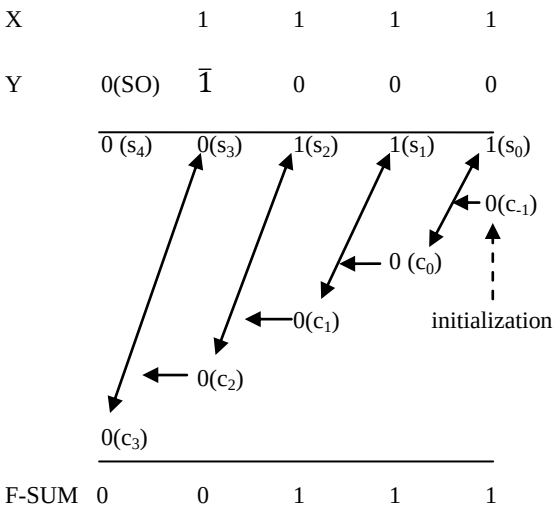

Fig. 1: Tracing the execution sequence for the reverse conversion of 100$\bar{1}$ using [7]

5. A NOTE ON [10] AS A DERIVATIVE OF [7]

The decomposition rules presented in [7] were exploited in [10] with a typical parallel prefix network [2] and the outcome was found to be significant. As the decomposition rules [7] have been found to be correct, [10] is also correct. The time complexity of [10] is Ω (log n) and later, some other RCSs for BSDN system ([5][6]) claimed to have the same time complexity following different approaches. But, so far, no unified study has been reported in the literature for the comparative merit assessment of [5][6][10] for RC of BSDN system as well as higher radix signed-digit number system (if extensible) in terms of area, delay, power and regularity of design as the parameters for investigation.

6. CONCLUSION

It has been shown that the rules on which [7] is based are correct, but, yet, they are not interoperable in such a way to ensure full-parallelism. This observation is compliant with the concept provided in [9]. As the decomposition rules presented in [7] are correct, [10] can be validated for achieving partial parallelism. Although, the asymptotic time requirement of [5][6][10] has been found to be the same; the full-fledged comparative study of their performances for RC of BSDN system and extensibilities for the RC of high-radix signed-digit numbers has not been reported in the literature. These problems will be studied as a part of the future research work of the authors.

7. REFERENCES

[1] Avizienis, A. 1961. Signed - digit number representation for fast parallel arithmetic, *IRE Trans on Electro Compu;* 10(3): 389 – 400.

[2] Parhami, B. 2009. Computer Arithmetic: Algorithms and Hardware Design. 1st ed. USA: Oxford Univ Press.

[3] Chakraborty, M. S. and Sao, S. K. 2014. Comments on area-time efficient sign detection technique for binary signed-digit number system proposed by Srikanthan et. al. IJCA, USA. 88(15): 38-40.

[4] Ruiz, G. A. and Granda, M. 2011. Efficient Canonic Signed Digit Recording. Microelect J. 42: 1090 – 1097.

[5] He, Y. and Chang, C-H. 2008. A Power - Delay Efficient Hybrid Carry - Lookahead/ Carry - Select Based Redundant Binary to Two's Complement Converter. IEEE Trans Circuits Syst I Regul Pap; 55(1): 336 – 346.

[6] Sahoo, S. K., Gupta, A., Asati, A. R. and Shekhar, C. 2010. A Novel Redundant Binary Number to Natural Binary Number Converter. J Signal Process Syst. 59: 297-307.

[7] Choo, I., Deshmukh, R. G. 2001. A novel conversion scheme from a redundant binary number to two's compliment binary number for parallel architectures. Proc IEEE SoutheastCon, Clemson, USA: 196 – 207.

[8] Wang, G. and Tull, M .P. 2004. A new Redundant Binary Number to two's complement Number Converter. Proc IEEE Region 5 Conf: ATLW, Norman, USA: 141-143.

[9] Blair, G. M. 1998. The Equivalence of Two's-Complement Addition and the Conversion of Redundant Binary to Two's-Complement Numbers. IEEE Trans Circuits Syst I Fundam Theory Appl; 45(6): 669-671.

[10] Veeramachaneni, S., Krishna, M. K., Avinash, L., Reddy, S. and Srinivas, M. B. 2007. High - Speed Redundant Binary to Binary Converter using Prefix Networks. Proc IEEE Intl Symp Circuits and Sys, Hyderabad, AP, India: 3271-3274