Asian Journal of Information Technology 4 (10): 901-906, 2005
© Grace Publications, 2005

The Conversion Algorithm and Implementation Between Carry-save and Binary Sign-digit Representations

Guoping Wang
Department of Engineering, Indiana University
Purdue University, Fort Wayne 2101 E. Coliseum. Blvd. Fort Wayne, IN 46805, USA

Abstract: The carry-propagation-free addition which is independently of operand length is feasible only if the outputs are expressed in the redundant representations. Binary sign-digit and carry-save representations are the two popular redundant formats which are widely used in the implementation of high-speed multipliers. In the previous studies, these two representations are treated separately. In this study, the algorithms and implementations are proposed for the conversion between them, thus, the computer arithmetic developed for one representation can be easily adapted to the other. The conversion overheads in the area and speed are also discussed.

Keywords: Binary signed-digit number, carry-save format, redundant representations, carry-propagation-free addition

INTRODUCTION

Constant-time binary addition is possible using redundant binary representations, among which the Binary Signed-Digit (BSD) and Carry-Save (CS) formats are well known. They have been used widely in the implementations of high-speed multipliers. For example[1-3], the high-speed multipliers were proposed using binary sign-digit implementations. Huang *et al.*[4] proposed a high-speed MAC unit using BSD representations. High-speed implementations of multipliers using carry-save representations were studied by Itoh *et al.*[5], Fadavi-Ardekani[6], Gustafsson *et al.*[7], Verma and Ienne[8], Junhyung and Taewhan[9], Ciminiera and Monthuschi[10]. These two formats are treated respectively in the previous researches. For example, the conversion algorithm and implementations from BSD into 2's-complement numbers were studied by Herrfelled and Hentschke[11], Walter[12], Montalvo and Parhi[13], Kornerup [14], Yen et al.,[15] and Phatak et al.,[16]. Lang and Bruguera proposed an algorithm and an implementation structure to detect the sign and overflow which can only be used in carry-save format representations. Srikanthan[18] proposed a sign detection technique only for binary sign-digit systems. In this study, the algorithm and structure of the conversion between CS and BSD representations are proposed, hence the algorithms which are developed for one representation can be easily adapted to be used for the other one.

BSD number system: Binary sign-digit representation is one of the Signed-Digit (SD) number systems originally introduced by Avizienis[19], which provides carry-propagation-free addition. In a signed-digit system, the individual digits have negative as well as positive values. Given a radix-(r) signed-digit number, each digit of the signed-digit number can take one of the following 2α+1 values:

$$\{-\alpha,\ldots,-1,0,1,\ldots,\alpha\} \qquad (1)$$

where the magnitude of a positive integer •must be within the following interval:

$$\left\lfloor \frac{r}{2} \right\rfloor \le \alpha \le r-1 \qquad (2)$$

The radix-2 signed-digit system (BSD) representation uses the digit set {-1, 0, 1} to represent binary numbers. The BSD number system is also called redundant because a given integer number may have more than one representation. For example, the radix-2 integer, $(7)_{10}$, can be represented in several ways, e.g., $[0\ 1\ 1\ 1]_{BSD}$, $[1\ 0\ 0\ -1]_{BSD}$, or $[1\ -1\ 1\ 1]_{BSD}$. Based on the BSD redundancy property, addition rules can be devised so that carry propagation is limited to only one digit position, thereby eliminating the possibility of a carry from the LSD (Least-Significant-Digit) to the MSD (Most-Significant-Digit). In a RB adder circuit implementation, the addition time is fixed and does not depend on the word length. Also, no explicit mechanism to handle the overall sign of a

Corresponding Author: Guoping Wang, Department of Engineering, Indiana University Purdue University, Fort Wayne 2101 E. Coliseum. Blvd. Fort Wayne, IN 46805, USA Tel: (260)481 6036 Fax: (260)481 6281

signed-digit number is required since it is determined by the most significant non-zero digit. Since the multiplication of two numbers is generally performed by the addition of partial products, the carry-propagation-free feature of the RB arithmetic can be used to design high-speed multipliers[1-3].

The conversion of 2's-complement to redundant binary: A limited precision BSD number, Δ, can be derived from the addition of a pair of *N*-bit 2's-complement numbers A and B[4].

$$\begin{aligned}(A+B)_{2C} &= A-(-B)_{2c}\\ &= A(\overline{B}+1)\\ &= A-\overline{B}-1\\ &= \left[-a_{N-1}2^{N-1}+\sum_{i=0}^{N-2}a_i2^i\right]+\left[\overline{b_{N-1}}2^{N-1}-\sum_{i=0}^{N-2}\overline{b_i}2^i\right]\overline{B}1\\ &= \left(-a_{N-1}+\overline{b_{N-1}}\right)2^{N-1}+\left[\sum_{i=0}^{N-2}(a_i-\overline{b_i})2^i\right]-1\\ &= \left[\delta_{N-1}2^{N-1}+\sum_{i=0}^{N-2}\delta_i2^i\right]-1\\ &= V-1\end{aligned} \tag{3}$$

where

$$\delta_{N-1}=-a_{N-1}+\overline{b_{N-1}},\delta=a_i-\overline{b_i}$$

for $0 \le I \le N-2$, 2c is the 2's-complement operations, is the 1's-complement operations, is the bit-complement.

The binary-signed digits can be encoded into binary in several ways. The binary signed digits{-1, 0, 0, 1} can be coded as positive/negative flag encoding {00, 01, 10, 11}, respectively, (Table 1).

Examining Eq. 3, beginning with the δ_i term, the signed digits are encoded using the relationship, $\delta_i=a_i-b_i$, where δ_i is a binary signed digit, $\delta \in \{-1, 0, 1\}$. The mapping equations for δ^-_i and δ^+_i are[4,20,21]

$$\begin{aligned}\delta^-_i &= a_i\\ \delta^+_i &= b_i \text{ for } 0 \le i \le N-2\end{aligned} \tag{4}$$

Similarly, in the Most Significant Digit (MSD) term of Equation, δ_{N-1} is encoded with the mapping equations

$$\delta^-_{N-1} = \overline{a_{N-1}}\ \delta^+_{N-1} = \overline{b_{N-1}} \tag{5}$$

The structure of mapping the sum of two 2's-complement binary numbers to a BSD number using positive/negative flag encoding (Fig. 1).

For example, a 2's-complement number $(00000101)_{2C}$ is converted directly into a BSD number $(01\ 01\ 01\ 01\ 01\ 11\ 01\ 11)_{BSD}$.

Table 1:Positive/Negative Flag Coding for BSD

α value	Encoded α^-	α^+
-1	0	0
0	0	1
0	1	0
1	1	1

Table2: Sign-Amplitude Coding for BSD

α value	Encoded α^S	$\alpha^\wedge$
-1	1	1
0	0	0
0	1	0
1	0	1

Table 3: Mapping table from a_i toδ_S δ_A

a_i	b_i	δ_i	δ_i^s	δ_i^A
0	0	1	0	1
0	1	0	don't care	0
1	0	0	don't care	0
1	1	-1	1	1

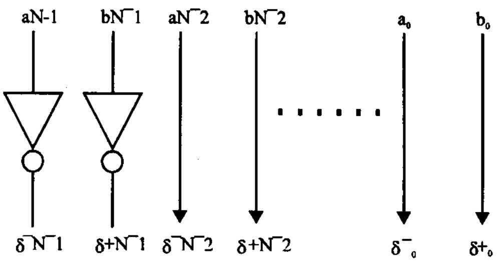

Fig.1: Mapping from binary numbers to a BSD for positive/negative encoding

The binary-signed digits can also be encoded into a sign-amplitude format[22], (Table 2).

Similarly, from Eq. 3 , Table 3 shows the mapping from a_i, b_i to δ_s,δ_A and the equations for $\delta_s$and δ_A using sign-amplitude encoding can be derived as

$$\begin{aligned}\delta_0^s = a_i,\ \delta_0^A = \overline{a_i \otimes b_i}\\ 0 \le 1 \le N-2\end{aligned} \tag{6}$$

The MSD term of Eq. 3, • is encoded with the mapping equations

$$\delta^s_{N-1} = \overline{a_{N-1}},\ \delta^A_{N-1} = \overline{a_{N-1} \oplus b_{N-1}} \tag{7}$$

Carry-save format representation: In the carry-save format, a binary number is represented in carry and save format. For example, the following carry-save redundant format as in Fig. 2 corresponds to the binary number 1100100.

The carry-save redundant representation format was used for high-speed multiplication implementation[23,24]. The Wallace-tree method is commonly used to realize high-speed multiplication[5-10] . The basic cell in carry-save addition is 3-to-2 or 4-to-2 CSA (Carry Save Adder), also called 3:2 or 4:2 counter. A 3:2 counter can be realized by

Table 4:Encoding of BSD Digits

BSD digits	Sign-amplitude Encoding ($signz^s$,$ampz^a$)	Positive/negative Encoding (z^+,z^-)
-1	11	01
0	00 or 10	00 or 11
1	01	10

Table 5:Conversion from Sign-Amplitude to Positive/Negative Encoding

BSD Digit	z^s	z^a	z^+	z^-
0	0	0	0	0
1	0	1	1	0
-1	1	1	0	1
invalid	1	0	x	x

Table 6: Conversion from Positive/Negative to Sign-Amplitude Encoding

BSD Digit	z^+	z^-	z^s	z^a
0	0	0	0	0
1	0	1	0	1
-1	1	0	1	1
invalid	1	1	x	x

```
 0 1 1 1 1 0
1 0 0 0 1 1 0
```

Fig. 2: A CS binary number example

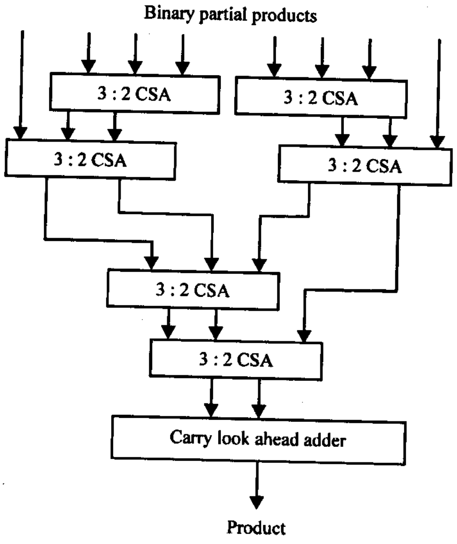

Fig.3: 3:2 Counter based 4×4 multiplier

a full adder, which reduces three numbers to two numbers. Fig. 3 and 4 are 4×4 multipliers, using 3:2 counters and 4:2 counters.

The traditional Wallace-tree method uses a 3:2 counter. This scheme results in a complicated interconnection between three-input/two-output counters. This makes the VLSI layout difficult and inefficient. The extended layout process increases the design complexity. As the multipliers increase in bit length, the interconnection becomes exponentially complicated. To solve this problem with conventional Wallace-trees, the 4:2 counters could be used instead of

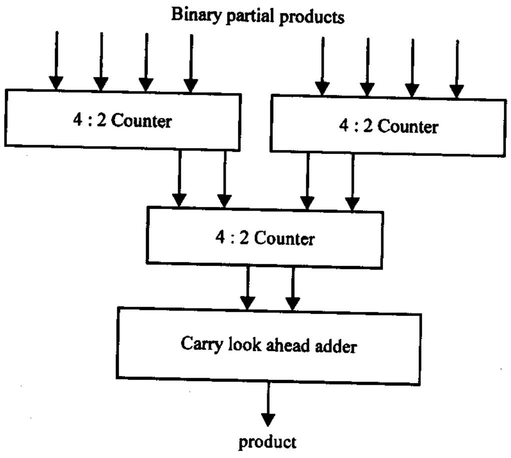

Fig. 4: 4:2 counter based 4×4 multiplier

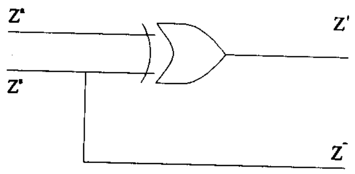

Fig. 5: BSD conversion from sign-amplitude to positive/negative encoding

3:2 counters. The use of 4:2 counters simplifies the interconnection drastically because the partial products are added using a binary tree.

BSD ENCODING AND CONVERSIONS

Digit sets{-1,0,1}are used in the BSD representations, thus, two bits are required to encode each BSD digit. There are two popular encodings for the BSD digits: sign-amplitude[22] and positive/negative flag encoding[1], (Table 4).

With the sign-amplitude encoding, the code 10 may be considered an alternate representation of 0 or else viewed as an invalid combination. In this study, 10 is considered as another representation of 0. This also applied to the encoding 11 for positive/negative flag representations. In the following, we look at the conversion algorithms between these two BSD encodings. Table 5 shows the conversion truth table from sign-amplitude to positive/negative flag encoding and Table 6 is the truth table for conversion from positive/negative flag encoding to sign-amplitude encoding. Equations 8 and 9 are the logic equations accordingly, while the logic diagram are shown in Fig 5 and 6.

$$Z^+ = Z^S \otimes Z^A \,, Z^- = Z^S \qquad (8)$$

$$Z^S = Z^+, Z^A = Z^+ \oplus Z^- \qquad (9)$$

CONVERSION FROM CS TO BSD

From Eq. 3, a BSD • can be considered as the sum of two 2's-complement *A* and *B* plus 1, which is, •, thus, a 3:2 counter can be used to obtain *A*+*B*+1 and . Equation 10 is the logic equation for full adder to add *x* and *y, cin* and get the sum: *s* and carry-out: *cout* to realize such 3:2 counter.

$$\begin{aligned} S &= X \oplus Y \oplus C_{in} \\ C_{out} &= XY + YC_{in} + XC_{IN} \end{aligned} \qquad (10)$$

Thus, the sum of *A*+*B*+1 can be obtained using a simplified 3:2 counter as shown in (Fig. 7).
Where *u* and *v* are the sum and carry format for *A*+*B*+1, and

$$\begin{aligned} &u_0 = \overline{a_0 \oplus b_0}\,, v_0 = 0 \\ &u_1 = a_i \oplus b_1,\ v_1 = a_o + b_o \\ &u_i = a_i \oplus b_i,\ v_i = a_{i-1}\ b_{i-1}\ (2 \le i \le N\text{-}1) \\ &u_N = u_{N-1} = a_{N-1} \oplus b_{N-1}, v_{N-1}\ b_{N-1} \end{aligned} \qquad (11)$$

U+V can be mapped into a BSD number according to Eq 4 and 5 or 6 and 7 depending on the BSD encodings: sign-amplitude or positive/negative flag encoding.

Figure 8 shows the diagram for the conversion from CS format into BSD.

CONVERSION FROM BSD TO CS

Consider the positive/negative flag encoding for the BSD representation, let

$$Z_{SD2} = \sum_{I=0}^{N-1} z_i 2^i$$

and we have

$$Z_i = Z_i^+ - Z_-^- \qquad (12)$$

therefore,

$$\begin{aligned} Z_{RB} &= \sum_{I=0}^{N-1} z_I 2^i = \sum_{I=0}^{N-1} \left(Z_i^+ - \overline{Z_i^-}\right) 2^i \\ &= \sum_{i=0}^{N-1} z_i^+\ 2^i - \sum_{i=0}^{N-1} \overline{z_i^-}\ 2^i \end{aligned} \qquad (13)$$

Let $Z^+ = \sum_{i=0}^{N-1} z_i^+ 2^i$ and $Z^+ = \sum_{i=0}^{N-1} z_i^+ 2^i$ then

$$Z_{BSD} = Z^+ - Z^- \qquad (14)$$

Table 7:Overhead of CS to BSD Conversion for N-bit Word Length

Total Number of Gates	Propagation Delay
2N	2 Gates

Table 8:Overhead of BSD to CS Conversion for N-bit Word Length

Total Number of Gates	Propagation Delay
2N	2 Gates

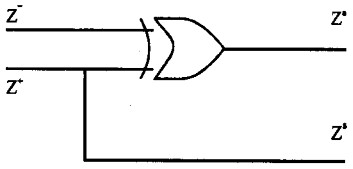

Fig. 6: BSD Conversion from Positive/Negative Flag to Sign-Amplitude Encoding

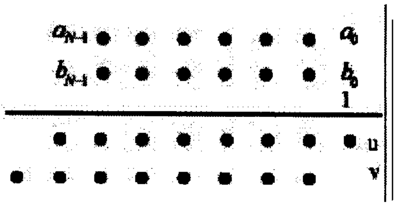

Fig. 7: Simplified 3:2 counter to realize A+B+1

where $Z^+$and Z^- are unsigned binary numbers, thus, •can be computed as

$$Z_{BSD} = Z^+ - Z^- = Z^+\ \overline{Z^-} + 1 \qquad (15)$$

where $\overline{Z^-}$ is the complement of Z^- with a sign flag '1' extension before the most significant bit. The simplified 3:2 counter structure in Fig. 7 and Eq. 11 can also be used for the computation of $Z^+ = +Z^- + 1$ to convert the Z_{SD2} BSD representation into a CS representation. For the sign-amplitude encoding for BSD, first Eq. 8 can be used to convert the sign-amplitude encoding into positive/negative flag encoding, then BSD can be converted into CS format according to Eq. 11 and Fig. 7. Figure 9 shows the diagram for conversion from BSD to CS format.

CONVERSION OVERHEAD

From Section 3, the conversion from CS into BSD involves two steps and the propagation delay for each step is only one gate which is independent upon the word length and *N* gates is required for each step for *N*-bit word length, so the total conversion overhead for this conversion is listed as Table 7.

Similarly, the overhead of BSD to CS conversion from Section is listed as Table 8.

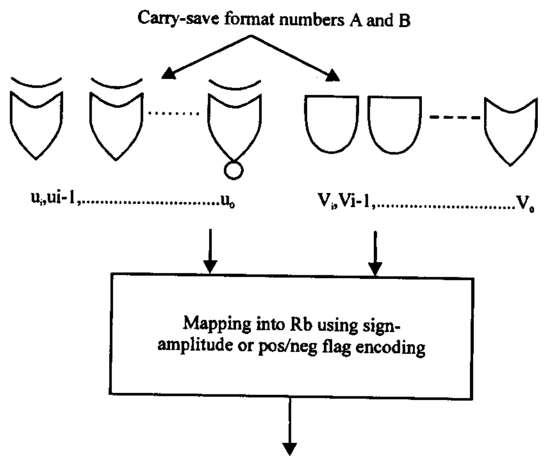
Carry-save format numbers A and B
u,ui-1,.............................u₀
V,Vi-1,.............................V₀
Mapping into Rb using sign-amplitude or pos/neg flag encoding

Fig. 8:Conversion Diagram from CS to BSD

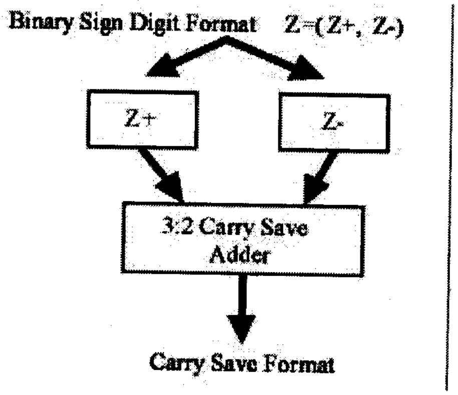

Fig. 9:Conversion Diagram from BSD to CS Format

EXAMPLE OF CONVERSION ALGORITHM APPLICATIONS

An area-time efficient sign detection technique for BSD was proposed by Srikanthan *et al.* The proposed method can be easily adapted to be used for carry-save format representation. The procedure is described here.

- Add 1 to the carry-save format according to Eq. 11, and this procedure adds only an extra gate delay and it is not dependent on the wordlength of the operands.
- Convert this carry-save result from step 1) to BSD format using Equation - and this conversion adds another gate delay. This delay is also independent of word length.
- Apply the sign-detection algorithm to this binary sign digit.

CONCLUSION

While the binary sign digit and carry-save representations are treated differently in the previous research, we have shown that the conversion algorithms and implementation between these two redundant representations without any carry-propagation delay problems. The important implications are that any advance in the algorithm and implementation in one redundant representation can be easily applied to the other.

REFERENCES

1. Takagi, N., H. Yasuura and S. Yajima, 1985. High-speed VLSI multiplication algorithm with a redundant binary addition tree IEEE Transaction on Computers, 34: 789-796.
2. Makino, H., Y. Nakase, H. Suzuki, H. Morinaka,.H. Shinohara and K. Mashiko, 1996. An 8.8-ns 54x54-bit Multiplier with high speed redundant binary Architecture. IEEE J. of Solid-state Circuits, 31: 773-783.
3. Vuillemin, J., 1983.A very fast multiplication algorithm for VLSI implementation, The VLSI J. Integration, 1: 39-52.
4. Huang, X., W. Liu and B.W.Y. Wei, 1994. A high-performance CMOS redundant binary Multiplication-and-Accumulation (MAC). IEEE Transactions on Circuits and Systems, 41: 33-39.
5. Itoh, N., Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, 2001. A 600-MHz 54×54-bit multiplier with rectangular-styled Wallace tree. IEEE J. Solid-State Circuits, 36: 249-257.
6. Fadavi-Ardekani, J., 1993. M×N Booth encoded multiplier generator using optimized Wallace trees. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1: 120-125.
7. Gustafsson, O., A.G. Dempster and L. Wanhammer, 2004. Multiplier blocks using carry-save adders. In: Proceedings of the 2004 International Symposium on Circuits and Systems, 2: 473-476.
8. Verma, A.K. and P. Ienne, 2004. Improved use of the carry-save representation for the synthesis of complex arithmetic circuits, In: ICCAD IEEE/ACM International Conference on Computer Aided Design, pp: 791-798.
9. Junhyung, U. and K. Taewhan. An optimal allocation of carry-save-adders in arithmetic circuits. IEEE Transactions on Computers, 50: 215-233.
10. Ciminiera, L. and P. Montuschi, 1996. Carry-save multiplication schemes without final addition. IEEE Transactions on Computers, 45: 1050-1055.

11. Herrfeld A. and S. Hentschke, 1995. Conversion of redundant binary into two's complement representations. Electronics Letters, 31: 1132-1133.
12. Walter, C.D., 1997. Analysis of delays in converting from a redundant representation. IEE Proc. Comput. Digit. Technol., 144: 219-221.
13. Montalvo, L.A. and K.K. Parhi, 1996. Radix-2 over-redundant digit set converters. In: 1996 IEEE International Symposium on Circuits and Systems, Atlanta, G.A., 4: 81-84.
14. Kornerup, P., 1994. Digit-set conversions: Generalizations and applications. IEEE Transactions on Computers, 43: 622-629.
15. Yen, S., C. Laih, C. Chen and J. Lee, 1992. An efficient redundant-binary number to binary number converter. IEEE J. Solid-State Circuits, 27: 109-112.
16. Phatak, D.S., T. Goff and I. Koren, 2001. Constant-time addition and simultaneous format conversion based on redundant binary representations. IEEE Transactions on Computers, 50: 1267-1278.
17. Lang T. and J.D. Bruguera, 1999. Multilevel reverse-carry computation for comparison and for sign and overflow detection in addition. In: Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp: 73-79.
18. Srikanthan, T., S.K. Lam and M. Suman, 2004. Area-time efficient sign detection technique for Binary signed-digit number system. IEEE Transactions on Computers, 53: 69-72.
19. Avizienis, A., 1961. Signed-digit number representations for fast parallel arithmetic. IRE Transactions on Electronic Computers, EC., 10: 389-400.
20. Shin K.W. and H.W. Jeon, 2000. High-speed complex-number multiplications based on redundant binary representation of partial products. Intl. J. Electronics, 87: 683-702.
21. Shin, K.W., B.S. Song and K. Bacrania, 1998. A 200-MHz complex number multiplier using redundant binary arithmetic. IEEE J. Solid-State Circuits, 33: 904-909.
22. Kuninobu, S., T. Nishiyama, H. Edamatsu, T. Taniguchi and N. Takagi, 1987. Design of high speed MOS multiplier and divider using redundant binary representation. In: Proceedings of 8th Symposium on Computer Arithmetic, pp: 80-86.
23. Wallace, C., 1964. A suggestion for a fast multiplier. IEEE Transactions on Electronic Computers, 13: 14-17.
24. Parhami, B., Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press.