# BSCB Full-Adder

Daniel Torno
Exorand Technology, France
contact@exorand.com

***Abstract*— This very short paper introduces a new implementation of a full-adder using the binary stored carry-or-borrow (BSCB) representation and the digit set {-1, 0, +1, +2}.**

***Keywords: full-adder; redundant binary representation; binary stored-carry-or-borrow representation.***

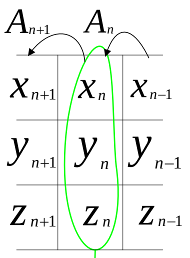
Let be three N-bit binary numbers denoted by X, Y and Z respectively:

$$X = x_{N-1}.2^{N-1} + x_{N-2}.2^{N-2} + ..... + x_1.2^1 + x_0.2^0$$
$$Y = y_{N-1}.2^{N-1} + y_{N-2}.2^{N-2} + ..... + y_1.2^1 + y_0.2^0$$
$$Z = z_{N-1}.2^{N-1} + z_{N-2}.2^{N-2} + ..... + z_1.2^1 + z_0.2^0$$

Let the sum be a (N+1) bit binary numbers denoted by S. For the sake of memory, table below shows the result of addition of 3 bits at position n leading to the full-adder based on carry-save form and the digit set {0, +1, +2, +3}.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $y_n$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $z_n$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $s_n$ | 0 | +1 | +1 | +2 | +1 | +2 | +2 | +3 |

In this computation there is no assumption about the possible value of a carry either in or out at position n. However, an assumption An=1 can be made at each position as displayed in figure below. The probability to generate a carry is equal to the probability not to generate one. New sum has also to handle an outgoing carry An+1.




$$s_n = x_n + y_n + z_n + A_n - 2 * A_{n+1}$$

With this assumption, sum is now expressed within the digit set {-1, 0, +1, +2}, this representation is called binary stored-carry-or-borrow (BSCB) by Parhami [1].

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $y_n$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $z_n$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $s_n$ | −1 | 0 | 0 | +1 | 0 | +1 | +1 | +2 |

Coding variables Un and Rn+1 expresses sum Sn.

| | | | | |
|---|---|---|---|---|
| $s_n$ | −1 | 0 | +1 | +2 |
| $r_{n+1}$ | 1 | 0 | 0 | 1 |
| $u_n$ | 1 | 0 | 1 | 0 |

Karnaugh table for Un:

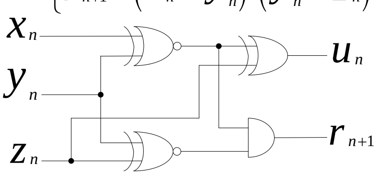
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $y_n$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $z_n$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $u_n$ | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

Karnaugh table for Rn+1:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $y_n$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $z_n$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $r_{n+1}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

A possible expression of sum variables and implementation with XOR and AND gates is given below.

$$s_n \begin{cases} u_n = \overline{x_n \oplus y_n \oplus z_n} \\ r_{n+1} = \left(\overline{x_n \oplus y_n}\right) \cdot \left(\overline{y_n \oplus z_n}\right) \end{cases}$$




Based on the BSCB representation, ripple carry adders, carry-look-ahead adders as well as array multipliers can be implemented (see [2]). It seems that the BSCB representation leads rather directly to XOR-AND-XOR gate implementation.

[1] Behrooz Parhami, "Generalized Signed-Digit Number Systems: A unifying Framework For Redundant Number Representation", *IEEE Transactions on Computer*, Vol 39, no. 1, pp 89-98, January 1990.

[2] Daniel Torno and Behrooz Parhami, "Arithmetic Operators Based on the Binary Stored-Carry-or-Borrow Representation," *Proc. 44th Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, pp.1148-1152, 7-10 November 2010.