Algorithms and multi-valued circuits for the multioperand addition in the binary stored-carry number system

D. Etiemble and K. Navi

LRI - UA 410 CNRS
Bat 490, Université Paris Sud
91405 Orsay Cedex, France

Abstract

We present the algorithms for the sum of two (three and four) digits in the Binary Stored-Carry number system by using the smallest set of values for the positional sum. We present the corresponding adders that use multivalued current-mode circuits. The implementation of multioperand additions using these adders is compared with the usual binary implementation.

1. Introduction

1.1. The binary stored-carry number system

The generalised signed-digit (GSD) number system has been defined [1] as a positional system with the digit set $\{-\alpha, -\alpha+1,..., \beta-1, \beta\}$ with the conditions $\alpha \geq 0$, $\beta \geq 0$ and $\alpha+\beta+1 > r$, where r is the number representation radix. The last condition restricts the definition to redundant number systems. The redundancy index of a GSD number system is $\rho = \alpha + \beta + 1 - r$. The redundant number systems have been considered for the implementation of the multioperand addition, as they have a carry-free or a carry-limited property [2-4].

The binary store-carry number system is a special case with $r = 2$, $\alpha = 0$, $\beta = 2$, $\rho = 1$, which has found wide applications in multioperand additions (multiplications). In this context, the usual unary encoding for BSC digits is {0,0} for 0, {0,1} for 1 and {1,1} for 2. Two unary-encoded BSC numbers can be added by a limited-carry circuit consisting of two levels of full adders.

1.2. Multivalued current-mode circuits

For many years, multivalued current-mode circuits have been considered as potential candidates to implement efficiently arithmetic functions [5-8]. As an example, a 2-µm current-mode CMOS 32 x 32 bit multiplier [9] has been demonstrated, with the same speed and half chip area and power dissipation compared with the best corresponding binary one at the same period. A detailed analysis of the properties of multivalued circuits can be found in [10].

Current-mode circuits operate on a limited number of current values, which are multiple of a current unit I. They are based on a set of operators: the analog sum of currents, the current sources, the current duplication to get multiple values of the current unit, and the current threshold detectors. These current-mode circuits can be implemented in many past or present technologies: CMOS, I^2L and CML-ECL with bipolar transistors, BiCMOS...

The tolerance issue has always been the Achilles' heel of current-mode circuits [11]. It is out-of the scope of the paper to present a detailed analysis of this problem. We just point out the basic aspects. To operate with m different current levels, m-1 threshold detectors are needed and the tolerance constraints are more and more difficult to satisfy when the number of current levels increase. Roughly, for many technologies, 8 levels is a limit. Multi-valued circuits are the most efficient (speed, complexity) with 3 or 4 current levels.

1.3. The limited-carry addition of BSC numbers

With the BSC number system, a carry-free addition is impossible. We briefly present the algorithm for limited-carry addition with the notation used in [1].

Let two numbers to be added have x_i and y_i as the i^{th} digits. In stage 1, for each position i, a position sum $p_i = x_i + y_i$ is computed and used to generate a range estimate e_{i+1} for the final transfer digit t_{i+1}. In stage 2, the position sum p_i and the range estimate e_i are used to compute a transfer digit t_{i+1} and an interim sum $w_i = p_i - r.t_{i+1}$. The final sum digit is $s_i = w_i + t_i$, which produce no new transfer. With BSC number system, $p_i \in \{0,1,2,3,4\}$, and the direct implementation needs 5-valued current-mode circuits. In [12], a multivalued CMOS implementation of the addition of BSC numbers is presented, based on the presented algorithm, with a 5-valued current-mode CMOS adder cell.

1.4. A limited number of current levels

To overcome the tolerance issue with multi-valued current mode circuits, we consider multivalued circuits with a limited number of current levels: 3 current levels, then 4 and 5 current levels. From the arithmetic point of

1063-6889/93 $03.00 © 1993 IEEE

view, it means that we only use an operational set of values $q_i \in \{0,1,..m\text{-}1\}$ where m is the number of current values that can be used. Obviously m-1 is smaller than the maximum value of p_i.

We consider several possible versions of the limited-carry algorithm according to m. We extend the problem to the sum of 3 digits and 4 digits. We compare the corresponding implementations of the multioperand addition with basic cells with the binary implementation using Wallace trees with Carry Save Adders (CSA) and modified Booth algorithm.

1.5. Notations and definitions

+ is the arithmetic sum on the digits of a defined set of digits. It will be implemented by the analog sum of currents, according to Kirchhoff laws. The dot is used for the logical product and ∨ is used for the logical sum.

Let E_n be the digit set $\{0, 1, ..., n\text{-}1\}$.

2. The BSC sum of 2 digits with $m = 3$

2.1. A new algorithm for the sum of 2 digits

As already mentioned, with BSC number system, $p_i \in E_5$ where p_i is the positional sum of two digits. If we limit the operational set to E_3, the positional sum cannot be achieved directly. Each BSC digit is first encoded according to binary components.

x_i is decomposed in a sum digit sum digit $x_i^{(0)}$ and a carry digit $x_i^{(1)}$ according to

$2\,x_i^{(1)} + x_i^{(0)} = x_i$

where $x_i^{(1)}, x_i^{(0)} \in E_2$ and $x_i \in E_3$.

The algorithm is the following one:

STEP 1 : Decompose x_i and y_i according to the binary components.

$2\,x_i(1) + x_i(0) = x_i$

$2\,y_i(1) + y_i(0) = y_i$

STEP 2 : Linearly add up p_{i1} and p_{i0} where p_{i1} and $p_{i0} \in E_3$

$p_{i1} = (x_i^{(1)} + y_i^{(1)})$

$p_{i0} = (x_i^{(0)} + y_i^{(0)})$

STEP 3: Decompose p_{i1} and p_{i0} according to the binary components.

$2\,p_{i1}^{(1)} + p_{i1}^{(0)} = p_{i1}$

$2\,p_{i0}^{(1)} + p_{i0}^{(0)} = p_{i0}$

Obviously, steps 1 to 3 correspond to the computation of the position digit p_i through binary decomposition and digit sum within E_3 according to the following expression:

$p_i = x_i + y_i = 2\,x_i^{(1)} + x_i^{(0)} + 2\,y_i^{(1)} + y_i^{(0)} = 2\,(x_i^{(1)} + y_i^{(1)}) + (x_i^{(0)} + y_i^{(0)})$

Table 1 gives the outputs $p_{i1}^{(1)}, p_{i1}^{(0)}, p_{i0}^{(1)}, p_{i0}^{(0)}$ corresponding to the significant inputs.

STEP 4: We decompose the range estimate e_{i+1} in two parts $c_{i+1}^{(2)}, c_{i+1}^{(1)}$ according to

$2\,c_{i+1}^{(2)} + 2\,c_{i+1}^{(1)} + w_i = p_i$ where $c_{i+1}^{(2)}, c_{i+1}^{(1)}, w_i \in E_2$

Table 1 also gives the outputs $c_{i+1}^{(2)}, c_{i+1}^{(1)}$ and w_i according to the positional sum p_i.

From Table 1

$c_{i+1}^{(2)} = p_{i1}^{(1)}$

$c_{i+1}^{(1)} = p_{i1}^{(1)} + p_{i1}^{(0)} + p_{i0}^{(1)}$

$w_i = p_{i0}^{(0)}$

STEP 5 : Linearly add up w_i and $c_i^{(1)}$

$v_i = w_i + c_i^{(1)}$

STEP 6 : Decompose v_i according to the binary components

$2\,v_i^{(1)} + v_i^{(0)} = v_i$

$v_i^{(1)}$ is the transfer digit t_{i+1}

STEP 7 : Linearly add up $v_i^{(0)}, t_i, c_i^{(2)}$

$s_i = v_i^{(0)} + t_i + c_i^{(2)}$

$t_i + c_i^{(2)} \in E_2$ because $c_i^{(2)} = 1$ only when $z_{i-1}=4$, which means $w_{i-1} = 0$, v_{i-1} = (0 or 1) and $v_{i-1}^{(1)} = t_i = 0$. As $t_i + c_i^{(2)} \in E_2$ and $v_i \in \{0, 1\}$, then $s_i \in E_3$.

Obviously, s_i depends on level i with output $v_i^{(0)}$, on level i-1 with $c_i^{(2)}$ and on level i-2 as t_i depends on $c_{i-1}^{(1)}$. The addition is propagation-limited.

$x_i + y_i$	$x_i^{(1)}$	$x_i^{(0)}$	$y_i^{(1)}$	$y_i^{(0)}$	p_{i1}	p_{i0}	$p_{i1}^{(1)}$	$p_{i1}^{(0)}$	$p_{i0}^{(1)}$	$p_{i0}^{(0)}$	$c_{i+1}^{(2)}$	$c_{i+1}^{(1)}$	w_i
0 + 0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 + 1	0	0	0	1	0	1	0	0	0	1	0	0	1
0 + 2	0	0	1	0	1	0	0	1	0	0	0	1	0
1 + 1	0	1	0	1	0	2	0	0	1	0	0	1	0
1 + 2	0	1	1	0	1	1	0	1	0	1	0	1	1
2 + 2	1	0	1	0	2	0	1	0	0	0	1	1	0

Table 1

2.2. Two-input BSC adders

Figure 1 shows the structure of the 2-digit BSC adder corresponding to the defined algorithm. "3BC" is the cell that implements the binary decomposition of ternary inputs used in STEPS 1, 3 and 6. From a circuit point of view, the "3BC" cell is a special case of the "*m*BC" cell, which is a *m*-valued current input to Binary current output Converter (mBC).

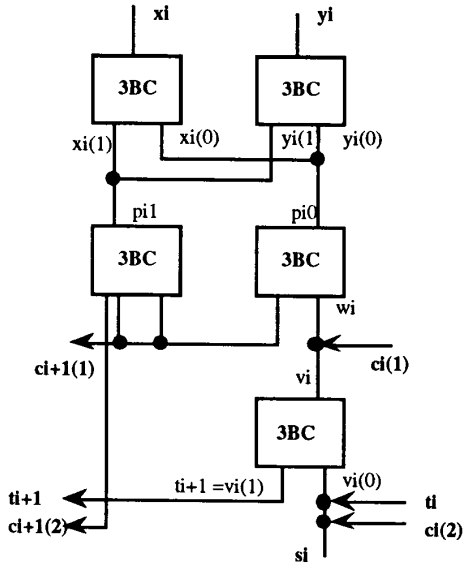

Figure 1: A 2-input BSC adder ("2IA") with "3BC" cells.

By using ternary current-mode circuits, the 2-digit BSC adder can be implemented with only one kind of cell: the "3BC" cell. It need 5 cells, and the propagation delay corresponds to 3 levels of "3BC" cells. Before comparing with the classical binary approach, we present a modified version of the 2-digit BSC adder.

A potential drawback of the 2-digit BSC adder is the 3 levels of "3BC" cells in the critical delay path. 3 of the 5 "3BC"s can be merged in an encoder circuit, as shown in Figure 2.

The outputs $v_i^{(1)}$, $v_i^{(0)}$, $c_{i+1}{}^{(1)}$ and $c_{i+1}{}^{(2)}$ can be derived from p_{i1} and p_{i0} (Table 2) by using the binary functions G (Greater) which have the following definition:

$x, i \in E_m$, $G_i(x) \in E_2$

$G_i(x) = 1$ iff $x > i$; $G_i(x) = 0$ otherwise.

The expressions are the following ones:

$c_{i+1}{}^{(2)} = G_1(p_{i1})$

$c_{i+1}{}^{(1)} = G_0(p_{i1}) \vee G_1(p_{i0})$

$v_i^{(1)} = c_i^{(1)} . G_0(p_{i0}) . \overline{G_1}\,(p_{i0})$

$v_i^{(0)} = \overline{c_i^{(1)}} . G_0(p_{i0}) . \overline{G_1}\,(p_{i0}) \vee c_i^{(1)} . (\overline{G_0}\,(p_{i0}) \vee G_1(p_{i0}))$

From a circuit point of view, it should be noticed that $v_i^{(1)}$, $v_i^{(0)}$ and $c_{i+1}{}^{(2)}$ correspond to currents, and $c_{i+1}{}^{(1)}$ is a voltage.

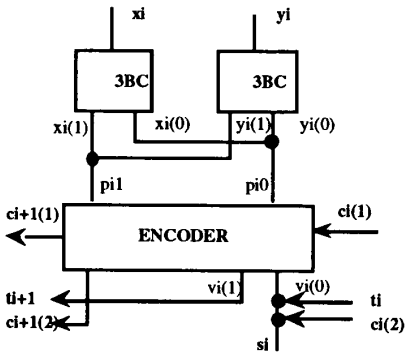

Figure 2: A modified 2-input BSC adder ("M2IA").

2.3. Multioperand addition

The n x n bit multiplication, which needs a multioperand addition is a good benchmark to compare the performance of the classical binary solution and the performance of the proposed approach with 2-input BSC adders. With the binary approach using a Wallace tree of carry save adders (CSA), a final carry-lookahead adder is needed to compute the two operands resulting from the Wallace tree. With BSC number representation, the final sum must be converted in the usual binary number representation by a similar lookahead adder.

For the comparison, we assume that Booth encoding is used with both approaches, which divides by 2 the initial number of operands. With BSC approach, the initial conversion from binary representation to BSC representation, which only need to add the binary digits, further divides by two the number of operands. We consider the 32 x 32 bit multiplication and the 64 x 64 bit multiplication.

We use the following definitions :

Let t_{d2} be the propagation delay of the CSA, t_{d3} the propagation delay of the "3BC" cell, and t_{enc3} the propagation delay of the encoder circuit ("3ENC") with m = 3. The propagation delay of the initial BSC 2-input adder is 3 t_{d3} and the propagation delay of the modified BSC 2-input adder is $t_{d3} + t_{enc3}$

In table 5 at the end of the paper, we give the overall comparison for all versions for the 32 x 32 bit multiplication. We give the propagation delay and the number of cells ("3BC"s or CSAs) for the multioperand addition (The number of cells per bit corresponds to the worst case, where all the operands are used in the multioperand addition). With m=3, the approach with BSC adders is faster only if $6\ t_{d2} > 9\ t_{d3}$, i.e. $t_{d3} < \frac{2}{3} t_{d2}$ or $t_{d3} + t_{enc3} < 2\ t_{d2}$ with modified BSC adders. Even if a precise comparison is technology-dependant and need to design the corresponding cells with defined Integrated Circuit Technologies, this condition can be hardly satisfied. If we assume a similar complexity (in term of number of transistors or chip area) for CSAs and "3BC", the multivalued approach uses more components than the binary one. Table 6 at the end of the paper gives the overall comparison for the 64 x 64 bit multiplication.

With Booth algorithm and a Wallace tree of CSAs, the number of CSA levels is $\lceil \log_{3/2}(n/2) \rceil$, where n is the number of operands to be summed. The base of the logarithm is the ratio between the number of inputs and outputs of the adder. With Booth algorithm and 2-input BSC adder, the number of adder levels is $\lceil \log_2(n/2) \rceil$. To increase the base of the logarithm, it is worthy to investigate the possibility to extend the number of inputs of the BSC adder, by using greater values of m

3. The sum of 3 BSC digits with $m = 4$

3.1. The algorithm of the sum of 3 BSC digits

We now consider 3 input digit x_i, y_i and z_i

$p_i = x_i + y_i + z_i$; $p_i \in E7$

$p_{i1} = x_i^{(1)} + y_i^{(1)} + z_i^{(1)}$; $p_{i1} \in E4$

$p_{i0} = x_i^{(0)} + y_i^{(0)} + z_i^{(0)}$; $p_{i0} \in E4$

p_{i1} and p_{i0} can be again decomposed into their binary components $p_{i1}^{(1)}$, $p_{i1}^{(0)}$, $p_{i0}^{(1)}$ and $p_{i0}^{(0)}$. But now, a 4-valued to binary current mode converter ("4BC") is needed.

$4\ c_{i+2}^{(2)} + 2\ c_{i+1}^{(1)} + w_i = p_i$ is the decomposition of p_i according to an intermediate digit w_i and two carry outputs $c_{i+2}^{(2)}$ and $c_{i+1}^{(1)}$. Now, $c_{i+1}^{(1)}$ correspond to a carry for the next digit, and $c_{i+2}^{(2)}$ is the carry for 2 digits further.

Table 2 gives the correspondence between the binary components of x_i, y_i , z_i and $c_{i+2}^{(2)}$, $c_{i+1}^{(1)}$, w_i.

From table 2, we can derive the following expressions:

$c_{i+2}^{(2)} = p_{i1}^{(1)} + p_{i1}^{(0)}.p_{i0}^{(1)}$

$c_{i+1}^{(1)} = p_{i1}^{(0)} + p_{i0}^{(1)}$

$w_i = p_{i0}^{(0)}$

Except for $p_{i1}^{(0)}.p_{i0}^{(1)}$, which need a logical product of the corresponding currents, only the analog sum + is needed.

For each digit, the intermediate sum w_i is added to $c_i^{(1)}$ and $c_i^{(2)}$.

$v_i = w_i + c_i^{(1)} + c_i^{(2)}$

v_i is decomposed again with a 4-valued to binary converter.

$v_i = 2\ v_i^{(1)} + v_i^{(0)}$, with $t_{i+1} = v_i^{(1)}$

The final digit s_i is the analog sum of $v_i^{(0)}$ and t_i from the previous stage.

$s_i = t_i + v_i^{(0)}$

Obviously, $s_i \in E3$.

3.2. Three-input BSC adders

The 3-input BSC adder corresponding to this algorithm is presented in figure 3. It uses 3 "3BC" and 3 "4BC" cells, plus a switch to implement an And function on currents. We call t_{d4} the propagation delay through the "4BC" cell. The propagation delay of the 3-input BSC adder is $t_{d3} + 2\ t_{d4}$.

p_i	p_{i1}	p_{i0}	$p_{i1}^{(1)}$	$p_{i1}^{(0)}$	$p_{i0}^{(1)}$	$p_{i0}^{(0)}$	$c_{i+2}^{(2)}$	$c_{i+1}^{(1)}$	w_i
0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	1
2	0	2	0	0	1	0	0	1	0
2	1	0	0	1	0	0	0	1	0
3	0	3	0	0	1	1	0	1	1
3	1	1	0	1	0	1	0	1	1
4	1	2	0	1	1	0	1	0	0
4	2	0	1	0	0	0	1	0	0
5	2	1	1	0	0	1	1	0	1
6	3	0	1	1	0	0	1	1	0

Table 2

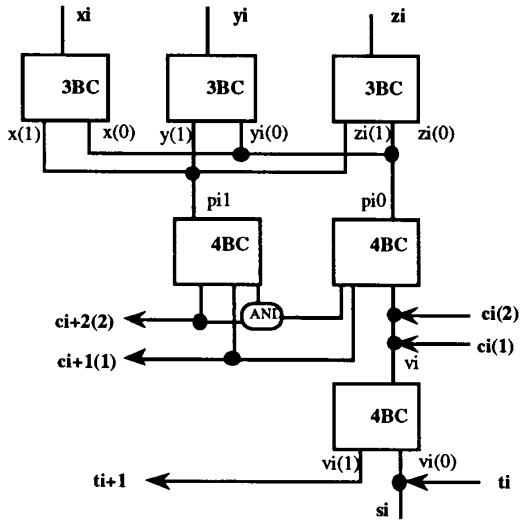

Figure 3: 3-input BSC adder with $m = 4$

Again, the two levels of "4BC" cells can be merged in an 4-valued encoder circuit, as shown in Figure 4.

According to table 2, the values of v_{i1} and v_{i0} can be derived from table 3.

$c_{i+1}^{(1)} = G1(p_{i0}) \vee G0(p_{i1}).\ \overline{G1}\ (p_{i1}) \vee G2(p_{i1})$

$c_{i+2}^{(2)} = G1(p_{i1}) \vee G0(p_{i1}).G1(p_{i0})$

$v_{i1} = c_i^{(2)} . c_i^{(1)} \vee (c_i^{(2)} \text{ xor } c_i^{(1)})$

$[G0(p_{i0}).\ \overline{G1}\ (p_{i0}) \vee G2(p_{i0})]$

$v_{i0} = (c_i^{(2)} \text{ nxor } c_i^{(1)}).[G0(p_i0).\ \overline{G1}\ (p_i0) \vee G2(p_i0)]$

$+ (c_i^{(2)} \text{ xor } c_i^{(1)}).\ [\ \overline{G0}\ (p_{i0}) \vee G1(p_{i0}).\ \overline{G2}\ (p_{i0})].$

For circuit implementation, v_{i1} and v_{i0} are currents and $c_{i+1}^{(1)}$ and $c_{i+2}^{(2)}$ are voltages.

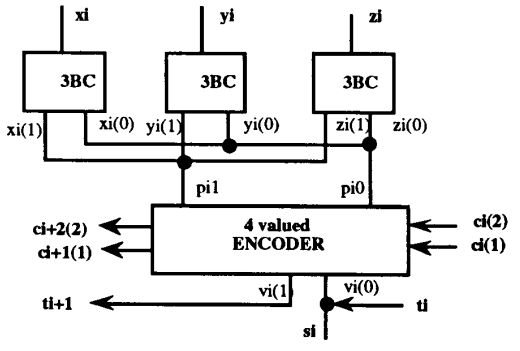

Figure 4: modified 3-input BSC adder

	p_{i0}	v_{i1}	v_{i0}
$c_i^{(2)}=c_i^{(1)}=0$	0	0	0
$c_i^{(2)}=c_i^{(1)}=0$	1	0	1
$c_i^{(2)}=c_i^{(1)}=0$	2	0	0
$c_i^{(2)}=c_i^{(1)}=0$	3	0	1
$c_i^{(2)}\neq c_i^{(1)}$	0	0	1
$c_i^{(2)}\neq c_i^{(1)}$	1	1	0
$c_i^{(2)}\neq c_i^{(1)}$	2	0	1
$c_i^{(2)}\neq c_i^{(1)}$	3	1	0
$c_i^{(2)}=c_i^{(1)}=1$	0	1	0
$c_i^{(2)}=c_i^{(1)}=1$	1	1	1
$c_i^{(2)}=c_i^{(1)}=1$	2	1	0
$c_i^{(2)}=c_i^{(1)}=1$	3	1	1

Table 3: Values of v_{i1} and v_{i0} according to p_{i0} and carry values

3.3. Multioperand addition

Again, we compare the binary and BSC implementations of the 32 x 32 bit and 64 x 64 bit multiplications. Both uses Booth algorithm.

Table 5 gives the propagation delay and the number of cells ("3BC"s, "4BC"s or CSAs) for the multioperand addition of the 32x32 multiplication. Table 6 gives the same information for the 64 x 64 bit multiplication.

4. The sum of 4 BSC digits with $m = 5$

4.1. The algorithm

It is a trivial extension of the previous algorithms, with four BSC digits called x_i, y_i, z_i and a_i.

$p_i = x_i + y_i + z_i + a_i\ ;\ p_i \in E9$

$p_{i1} = x_i^{(1)} + y_i^{(1)} + z_i^{(1)} + a_i^{(1)}\ ;\ p_{i1} \in E5$

$p_{i0} = x_i^{(0)} + y_i^{(0)} + z_i^{(0)} + a_i^{(0)}\ ;\ p_{i0} \in E5$

p_{i1} and p_{i0} can be again decomposed into their binary components $p_{i1}^{(1)}$, $p_{i1}^{(0)}$, $p_{i0}^{(1)}$ and $p_{i0}^{(0)}$. But now, a 5-valued to binary current mode converter ("5BC") is needed.

$4\ c_{i+2}^{(3)} + 4\ c_{i+2}^{(2)} + 2\ c_{i+1}^{(1)} + w_i = p_i$ is the decomposition of p_i according to an intermediate digit w_i and three carry outputs $c_{i+2}^{(3)}$, $c_{i+2}^{(2)}$ and $c_{i+1}^{(1)}$. Now, $c_{i+2}^{(3)}$ and $c_{i+2}^{(2)}$ correspond to a carry for two digits further, and $c_{i+1}^{(1)}$ is the carry for next digit.

Table 4 gives the correspondence between the binary components of p_i and $c_{i+2}^{(3)}$, $c_{i+2}^{(2)}$, $c_{i+1}^{(1)}$ and w_i.

p_i	p_{i1}	p_{i0}	$p_{i1}^{(2)}$	$p_{i1}^{(1)}$	$p_{i1}^{(0)}$	$p_{i0}^{(2)}$	$p_{i0}^{(1)}$	$p_{i0}^{(0)}$	$c_{i+2}^{(3)}$	$c_{i+2}^{(2)}$	$c_{i+1}^{(1)}$	w_i
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0	1
2	0	2	0	0	0	0	1	0	0	0	1	0
2	1	0	0	0	1	0	0	0	0	0	1	0
3	0	3	0	0	0	0	1	1	0	0	1	1
3	1	1	0	0	1	0	0	1	0	0	1	1
4	0	4	0	0	0	1	0	0	1	0	0	0
4	1	2	0	0	1	0	1	0	0	1	0	0
4	2	0	0	1	0	0	0	0	0	1	0	0
5	1	3	0	0	1	0	1	1	0	1	0	1
5	2	1	0	1	0	0	0	1	0	1	0	1
6	2	2	0	1	0	0	1	0	0	1	1	0
6	3	0	0	1	1	0	0	0	0	1	1	0
7	3	1	0	1	1	0	0	1	0	1	1	1
8	4	0	1	0	0	0	0	0	1	1	0	0

Table 4

$c_{i+2}^{(3)} = p_{i1}^{(2)} + p_{i0}^{(2)}$

$c_{i+2}^{(2)} = p_{i1}^{(2)} + p_{i1}^{(1)} + p_{i0}^{(1)}$

$c_{i+1}^{(1)} = p_{i1}^{(0)} + p_{i0}^{(1)}$

$w_i = p_{i0}^{(0)}$

For each digit i, w_i is added to $c_i^{(1)}$, $c_i^{(3)}$ and $c_i^{(2)}$

$u_i = w_i + c_i^{(1)} + c_i^{(2)} + c_i^{(3)}$

u_i is decomposed again with a 5-valued to binary converter.

$u_i = 4\,u_{i+2}^{(2)} + 2\,u_{i+1}^{(1)} + u_i^{(0)}$

$v_i = u_i^{(2)} + u_i^{(1)} + u_i^{(0)}$

v_i is decomposed with a 4-valued to binary converter, and $v_i(1) = t_{i+1}$

The final digit s_i is the analog sum of $v_i^{(0)}$ and t_i from the previous stage.

$s_i = t_i + v_i^{(0)}$

Obviously, $s_i \in \{0, 1, 2\}$.

4.2. Four-input BSC adders with m=5

The corresponding circuit is given in figure 5. It uses 4 "3BC", 3 "5BC", and 1 "4BC". Its major drawback is that the propagation delay is $t_{d3} + 2\,t_{d5} + t_{d4}$, where t_{d5} is the propagation delay through the "5BC" cell. It is also possible to define an 5-valued encoder circuit to replace the three "5BC" and the final "4BC" cells, but it is quite evident that the corresponding circuit will be too complicated to get a small propagation delay through this circuit (t_{enc5}).

Two-input BSC adder with m=5

With m = 5, a direct implementation of the two-input BSC adder, which corresponds to the straightforward algorithm, is possible. It has been presented in [12].

$p_i = x_i + y_i$, where $x_i, y_i \in E_3$ and $p_i \in E_5$.

Compute $v_i^{(0)}$ and $t_{i+1} = v_i^{(1)}$ as functions of $G_j(p_i)$ and t_i.

$s_i = v_i^{(0)} + t_i$

We call the corresponding circuit "5-2I-adder"

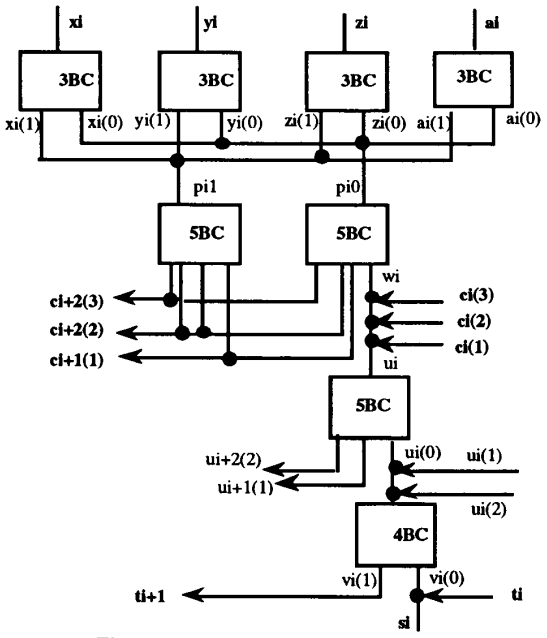

Figure 5: Four-input BSC adder

4.3. Multioperand addition

Table 5 gives the propagation delay and the number of cells ("3BC"s, "4BC"s or CSAs) for the multioperand addition of the 32x32 bit multiplication. Table 6 gives the same information for the 64 x 64 bit multiplication.

5. Concluding remarks

Multivalued current-mode circuits have often been presented as a potential candidate for efficient VLSI integration of arithmetic circuits. However, the tolerance issue which is a potential drawback of multivalued circuits, must be considered. To overcome this issue, we have presented algorithms for the sum of BSC numbers that uses the minimum number of current levels.

We have shown that the sum of two BSC numbers can be implemented with current mode circuits, by using only one kind of cell: the 3-valued to binary current-mode converter ("3BC"). This can be extended to the sum of three (resp. four) BSC number that can be implemented by using "3BC", "4BC" (resp. "5BC") cells. For efficiency, these BSC adders can be modified by defining encoder circuits, that reduce the number of "gate" levels to implement.

	Propagation delay	Number of components/bit
Booth + Wallace tree of CSAs	6 levels 6 t_{d2}	14 CSAs
2-input BSC adders with m = 3	3 "2IA" 9 t_{d3}	7 "2IA" 35 "3BCs"
modified 2-input BSC adders with m = 3	3 "M2IA" 3 t_{d3} + 3 t_{enc3}	7 "M2IA" 14 "3BCs" + 7 "3ENC"
Booth + 3-input BSC adder with m = 4	2 "3IA" 2 t_{d3} + 4 t_{d4}	3 "3I-adder" + 1 "2I-adder" 14 "3BC" + 9 "4BC"
Booth + modified 3-input BSC adder with m = 4	2 "3IA" 2 t_{d3} + 2 t_{enc4}	3 "3I-adder" + 1 "2I-adder" 11"3BC" + 3 "4ENC" + 1 "3ENC"
Booth + 4-input BSC adders	1 "4IA" + 1 "2IA" 4 t_{d3} + 2 t_{d5} + t_{d4}.	2 "4I adder" + 1 "2I adder" 13 "3BC" + 2 "4BC" + 6 "5BC"
Booth + modified 4-input BSC adders	1 "4IA" + 1 "2IA" 2 t_{d3} + t_{enc5} + t_{enc3}.	2 "4I adder" + 1 "2I adder" 7 "3BC" + 1 "4ENC" + 1 "5ENC"
Booth + 2-input BSC adders with m=5	3 "5-2IA" 3 $t_{5\text{-}2IA}$	7 "5-2IA"

Table 5: 32-operand addition.

	Propagation delay	Number of components/bit
Booth + Wallace tree of CSAs	8 levels 8 t_{d2}	30 CSAs
2-input BSC adders with m = 3	4 "2IA" 12 t_{d3}	15 "2IA" 75 "3BCs"
modified 2-input BSC adders with m = 3	4 "M2IA" 4 t_{d3} + 4 t_{enc3}	15 "M2IA" 30 "3BCs" + 15 "3ENC"
Booth + 3-input BSC adder with m = 4	2 "3IA" + 1 "2IA" 5 t_{d3} + 4 t_{d4}	7 "3I-adder" + 1 "2I-adder" 26 "3BC" + 21 "4BC"
Booth + modified 3-input BSC adder with m = 4	2 "3IA" + 1 "2IA" 3 t_{d3} + 2 t_{enc4} + t_{enc3}	7 "3I-adder" + 1 "2I-adder" 23 "3BC" + 7 "4ENC" + 1 "3ENC"
Booth + 4-input BSC adders	2 "4IA" 2 t_{d3} + 4 t_{d5} + 2t_{d4}	5 "4I adder" 20 "3BC" + 5 "4BC" + 15 "5BC"
Booth + modified 4-input BSC adders	2 "4IA" 2 t_{d3} + 2 t_{enc5}	5 "4I adder" 20 "3BC" + 5 "5ENC"
Booth + 2-input BSC adders with m=5	4 "5-2IA" 4 $t_{5\text{-}2IA}$	15 "5-2IA"

Table 6: 64-operand addition.

By considering the multioperand addition that is needed in 32 x 32 bit and 64 x 64 bit multiplications, we have set up the basis for a VLSI comparison of the usual binary implementation (Wallace tree of CSAs) with a multivalued current mode implementation of BSC adders. This detailed comparison is out of the scope of this paper. It involves the delay (Tables 5 and 6 gives the terms of the delay comparison). It involves the complexity, in term of chip area and power dissipation. Only a VLSI design of the different cells in various technologies (i.e. CMOS and bipolar ECL at least) can give the figures for this complexity comparison. In appendix, we present the electrical schemes of the basic BSC cells with CMOS and ECL technologies that we are considering for VLSI implementation.

6. References

[1] B. Parhami, "Generalised Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations", in IEEE Trans. Comp., Vol. 39, N° 1, January 1990.

[2] A. Azivienis, "Signed-Digit Number Representations for fast Parallel Arithmetic", in IRE Trans. Elect. Comp., EC-10, pp. 389-400, Sept. 1961.

[3] N. Tanaki, H. Yasuura and S. Yajima "High Speed VLSI multiplication algorithm with a redundant binary addition tree", in IEEE Trans. Comp., Vol. .34, N° 9, pp 789-796, Sept. 85.

[4] J.E. Vuillemin "A Very Fast Multiplication Algorithm for VLSI Implementation", Integration, VLSI J. 1, 1, pp. 39-52, Apr. 1983.

[5] T.T.Dao, "Threshold I^2L and its Application to Binary Symmetric Functions and Multivalued Logic", IEEE J. Solid-State Circuits, vol. SC-12, pp 463-472, Oct. 1977.

[6] S.P. Onneweer and H.G. Kerkhoff, "Current-Mode High Radix Circuits", Proc. Int'l. Symp. Multiple Valued Logic, pp. 60-69, May 1986

[7] S. Kawahito, M. Kameyama and T. Higuchi,, "VLSI-Oriented Bi-Directional Current Mode Arithmetic Circuits Based on the Radix-4 Signed Digit Number System", in Proc. Int'l Symp. Multiple Valued Logic, pp 70-77, May 1986

[8] K.W. Current, D.A. Freitas and F.A. Edwards, "CMOS quaternary threshold logic full adder circuits ", in Proc. Int'l Symp. Multiple Valued Logic, pp 318-322, May 1985.

[9] S. Kawahito, M. Kameyama, T. Higuchi, H. Yamada, "A 32 x 32 bit Multiplier Using Multiple-Valued MOS Current-Mode Circuits", IEEE J. Solid-State Circuits, vol. SC-23, pp 124-132, Feb. 1988

[10] D. Etiemble and M. Israël, "Comparison of binary and multivalued integrated circuits according to VLSI criteria", IEEE Computer, April 1988.

[11] D. Etiemble, "On the performance of multivalued integrated circuits: Past, Present and Future", Proc. Int'l. Symp. Multiple Valued Logic, pp 156-164., May 1992

[12] S. Kawahito, Y. Mitsui, M. Ishida and T. Nakamura, "Parallel Hardware Algorithms with Redundant Number Representations for Multiple-Valued Arithmetic VLSI", in Proc. Int'l Symp. Multiple Valued Logic, pp 337-345, May 1992.

[13] C.L.Chen, "2.5-V Bipolar CMOS Circuits for 0.25μm BiCMOS Technology", IEEE J. Solid-State Circuits, vol. SC-27 pp 485-491, Apr. 1992

7. Appendix: Examples of BSC cells in CMOS and ECL technologies.

Figure 6 shows the circuit diagram of the "3BC" cell in CMOS technology. Figure 7 presents the corresponding ECL circuit diagram. The simulations and VLSI implementation of the sets of multivalued circuits are being done, using a 1.2μm BiCMOS technology to evaluate the real performance of both CMOS and ECL approaches.

Preliminary simulation results shows that it is difficult to get less than 6 ns for the propagation delay of the 2-input BSC adder when using current mode CMOS circuits. With current mode ECL circuits without using series-gating, the critical path delay for the 2-input BSC adder is less than 1.5 ns with 0.4mA current switchs. This result is interesting when considering that advanced BiCMOS technologies with a 2.5V power supply will no longer use series gating in the future [13]. In such a BiCMOS technology, ECL 2-input BSC adders could exhibit a speed advantage versus binary CSAs for multioperand additions.

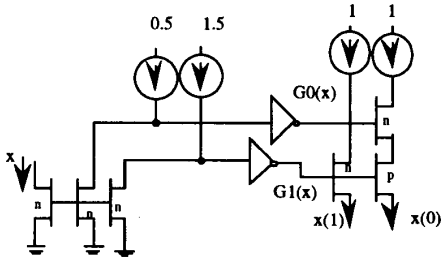

Figure 6: "3BC" cell in current mode CMOS technology.

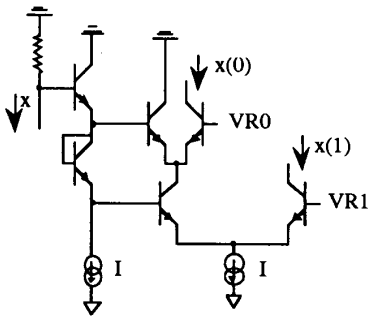

Figure 7: "3BC" cell in current mode ECL technology