Abstract. A fast parallel arithmetic using a Cellular Automaton (CA) environment is presented. The Redundant Binary (RB) number representation, first studied for optical computing, is used in order to perform a carry-free parallel addition or algebraic sum of arbitrary large numbers in constant time.

1. Introduction

Parallel computing models, like the Cellular Automaton (CA), that explicitly consider the ultimate impact of fundamental physical limitations, have been the subject of several recent studies (e.g., [1, 3]). Two-dimensional CAs are inherently parallel computing machines, that are very suitable for processing two-dimensional data structures in a way similar to that used on optical computers. Both devices can be considered as finite state machines that perform operations on two-dimensional data wavefronts in the finite time interval $[t_n, t_{n+1}]$, where $t_n$ is a transition between two machine states.

A method for performing binary additions on CAs has been presented by Sheth et al. in [5]. This method works serially and the addition of two numbers is performed in $O(N)$ time, where $N$ is the length of the bit strings representing the operands. The method of Sheth et al. is similar to that presented by Huang et al. [8] to implement a binary arithmetic on optical computers. As in the case of optical computing, a faster arithmetic can be obtained using suitable number representations [6, 7, 9, 10, 12]. In this paper, a carry-free algebraic sum on CAs is implemented. This operation can be performed in constant time independently of the length of the bit strings representing the operands, using the Redundant Binary (RB) number representation introduced in [12].

The attainment of parallel carry-free addition using redundant number representations has been investigated by many authors. Using this approach it is possible to build totally parallel adders operating in constant time (i.e.,
the adding time is independent of the operand digit string length \(N\), using very small truth tables which are independent of the digit position. The modified signed digit (MSD) representation is the one most widely investigated, particularly in the field of optical computing, on which there are numerous works (e.g., [10, 11]). In some recent works the RB number representation has been presented and studied in detail (e.g., [12, 13]). This number representation has the following advantages: it allows building an inherently parallel arithmetic with a two step carry-free algebraic sum, it naturally fits the 2s complement binary number system, and it requires only two symbols \{0, 1\} instead of three or more (such as required by the MSD).

2. The Redundant Binary number representation

As presented in [12], an unsigned integer \(x\) is in Redundant Binary representation (RB representation) when:

\[
x = \sum_{i=0}^{N-1} a_i 2^{i - \lfloor \frac{i}{2} \rfloor}
\]

with \(N\) even

(1)

where \(a_i \in \{0, 1\}\), \(i\) is the position index, \(N\) is the length of the bit string, and the most significant bit is on the left end of the bit string. The symbols \(\lfloor \rfloor\) represent rounding up to the next integer. In the RB representation each number has a canonical form and several redundant representations (e.g., [12]).

Informally, the RB number representation is obtained by doubling the weight positions in the natural binary representation of a given number \(x\); in this way, a sequence of \(N = |x|\) bit pairs \((n_{N-1}, r_{N-1}), \ldots, (n_0, r_0)\) is generated, with position weight \(2^{N-1}, 2^{N-2}, \ldots, 2^0, 2^0\), respectively. In each pair the bits have the same weight, the left and right bit are called the \(n\) (normal) and \(r\) (redundant) bit, respectively.

In a way similar to the 2s complement number system, signed RB numbers can also be defined (e.g., [12]):

\[
x = - \sum_{i=N-2}^{N-1} a_i 2^{i - \lfloor \frac{i}{2} \rfloor} + \sum_{i=0}^{N-3} a_i 2^{i - \lfloor \frac{i}{2} \rfloor}
\]

with \(N\) even.

(2)

From equations 1 and 2 it follows that an RB representation of a number can be obtained from its binary (or 2s complement) representation by using the following one step recoding rules:

\[
0 \rightarrow 00 \quad 1 \rightarrow 10
\]

wherein \(n_i\) bits take the same value of the corresponding binary bit of the same weight, while all \(r_i\) bits are zeroed.

The decoding of an RB number can be performed simply by one binary addition. In fact, the decoded value is the sum between two binary numbers, the first made by the \(n_i\) digits and the second by \(r_i\) digits. The nonconstant
3. The cellular automaton local rule

The formal description of the CA local rule performing the parallel algebraic sum will be given using the CAM-FORTH language of the Cellular Automata Machine (CAM) designed by the Information Mechanics Group of the Massachusetts Institute of Technology (e.g., [2, 4]). However, for the sake of clarity, the normal order notation is preserved instead of using the Reverse
Table 1: Symbolic substitution rule table for the algebraic sum of RB numbers. This table acts on \(nr\) pairs; \(u\) and \(l\) indicate the upper and lower row respectively. The lower output pair is shifted left one position.

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>10</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>10</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

Polish Notation adopted by the CAM-Forth, and the simple construct: \(\text{begin} \ldots \text{end}\) is also used. According to the CAM terminology, the state binary components of cells will be denoted as \(\text{Planes}\). In particular, the cell state of the local rule consists of four \(\text{Planes}\) and the adopted neighborhood is the \(\text{Moore}\) neighborhood.

A pair of RB input numbers is located on any pair of adjacent rows (which will be respectively denoted as \(\text{upper}\) and \(\text{lower}\) row), of the initial configuration of Plane 0. Consequently, if the size of the CA is \(N\) it is then possible to perform \(N/2\) RB algebraic sums in parallel. The output can be recovered on Plane 0 of the lower row after two applications of the local rule consisting of two consecutive \(\text{phases}\). In the first phase the rule table is applied without considering the shifting of the resulting lower row. The left shift of the lower row is then performed in the second phase. After two applications of the local rule (i.e., after four phases), the resulting sum is obtained on the lower row.

\[
\begin{align*}
(1703865068)_{10} & \quad (10111110110010010000000100110001001001000001111011001110100000)_{RB} \\
(-1876663285)_{10} & \quad (111010111001011101010010000011100110010100101110010010)_{RB} \\
(1820616592)_{10} & \quad (001010001010000101000000010100000010100000001000000000000)_{RB} \\
(-1993414809)_{10} & \quad (110101100101101000010010010100100101100110110100111001010)_{RB} \\
(0)_{10} & \quad (00000000000000000000000000000000000000000000000000000000000000000000000)_{RB} \\
(-172798217)_{10} & \quad (010101001101001010001100011001100110010110110011001011010101010)_{RB} \\
\end{align*}
\]

Figure 2: Algebraic sum of two signed RB numbers. After two applications of Table 1 the upper row is zeroed while the lower row contains the algebraic sum.
The other three Planes assume the following roles: Plane 1 is used to distinguish between the two phases, Plane 2 is devoted to distinguishing between the left and right cells (i.e., between the bit of type $n$ and that of type $r$), and Plane 3 is used to distinguish between the upper and lower operand. The starting configuration is defined in the following way: the RB operands are located on Plane 0, all bits of Plane 1 are set to 1, and each row of Plane 2 is an alternating sequence of 0/1 bits (starting with 1). Finally, Plane 3 has the bits of the upper operand set to 1 and those of the lower operand set to 0.

Here is the CAM-Forth representation of the CA local rule.

```forth
new-experiment
n/moore & /centers
RB_algebraic_sum:           {* Rules for Plane 0 }
begin
  sum: 2 * & center’ + & center;
  if center’                        {* Computing phase }
    then begin
      if sum = 0
        then if not(n.west)
          then (west xor center) > pln0;
          else not(west xor center) > pln0;
        if sum = 1
          then if north
            then (center and east) > pln0;
            else not(east) > pln0;
        if sum = 2
          then false > pln0;
        if sum = 3
          then east > pln0;
        end;
      else if not(& center’)
        then east > pln0;
        else center > pln0;
      not(center’) > pln1;            {* Shifting phase }
      & center > pln2;
      & center’ > pln2;
    end.
make-table algebraic-sum

& center > pln3;

In Figure 3 the execution of the local rule on a CA of size $N = 64$ working on 32 pairs of RB numbers simultaneously is shown.

4. Conclusion

A CA performing constant time carry-free parallel addition or algebraic sum has been presented. The resulting CA can be implemented using the CAM
Figure 3: Execution of the local rule on a CA of size $N = 64$.

A property of the RB number algebraic sum, performed with the rules of Table 1, is that the algebraic sum of two strings, considered as the concatenation of many RB numbers, gives a resulting string that is the concatenation of RB sums (e.g., [12]). This property is very useful if CAs with large $N$ are used.
References


