Complex Systems **8** (1994) 435–441

Fast Parallel Arithmetic on Cellular Automata

A. Clementi
G. A. De Biase
A. Massini
Dipartimento di Scienze dell'Informazione,
Università di Roma "la Sapienza",
Via Salaria 113 00198 - Roma, Italy

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:

$$x = \sum_{i=0}^{N-1} a_i 2^{i-\lceil \frac{i}{2} \rceil} \qquad \text{with } N \text{ even} \tag{1}$$

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 $\lceil \rceil$ 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 $\langle n_{N-1}, r_{N-1} \rangle, \ldots, \langle n_0, r_0 \rangle$ is generated, with position weight $2^{N-1}, 2^{N-1}, \ldots, 2^0, 2^0$, 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]):

$$x = -\sum_{i=N-2}^{N-1} a_i 2^{i-\lceil \frac{i}{2} \rceil} + \sum_{i=0}^{N-3} a_i 2^{i-\lceil \frac{i}{2} \rceil} \qquad \text{with } N \text{ even.} \tag{2}$$

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:

$$0 \rightarrow 00 \qquad 1 \rightarrow 10$$

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

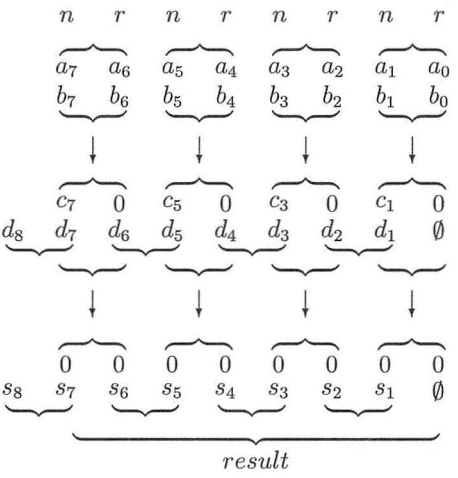

Figure 1: Parallel application of the algebraic sum rule (Table 1) on two RB numbers. a_i and b_i are the bits of the input operands, c_i and d_i are the bits of the intermediate sum, and s_i are the bits of the result.

time complexity in the decoding operation is not particularly important because this operation is required only when data must be given to the external world.

In [12] it is proved that the RB representation of numbers permits a two step totally parallel algebraic sum performable by a rule table which acts on $\langle n_i, r_i \rangle$ bit pairs. This rule table operates on two superposed and aligned sequences representing the RB input numbers (simply denoted as *upper* and *lower* number, see Figure 1) and gives results that are still on two superposed and aligned sequences.

Table 1 shows this rule, which is twice applied in parallel on all bit pairs of two RB numbers (operands) giving the result. The upper number is zeroed and the lower one gives the algebraic sum. The rule table does not depend on the relative position of the bit pair in the sequences, however, it has an unavoidable nonhomogeneous structure: it depends both on which RB input number (lower or upper) the pair belongs to and on the type of the bit (n_i or r_i). An example of an RB algebraic sum is shown in Figure 2.

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.

u / l	00	01	10	11
00	00 00	10 00	00 01	10 01
01	00 01	10 01	00 10	10 10
10	00 01	10 01	00 10	10 10
11	00 10	10 10	00 11	10 11

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

A pair of RB input numbers is located on any pair of adjacent rows (which will be respectively denoted as *upper* and *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 *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.

$(1703865068)_{10}$ $(10111110110100100100000010011000100101000001111101100011101000 00)_{RB}$
$(-1876663285)_{10}$ $(11101011100101101100100000000101100001110110010101001011110001010)_{RB}$

$(1820616592)_{10}$ $(00101000101000001000000000100000001010000010101010000010000000 00)_{RB}$
$(-1993414809)_{10}$ $(11010111001011010000100010010100100100110001010011001111001010 10)_{RB}$

$(0)_{10}$ $(00000000000000000000000000000000000000000000000000000000000000 00)_{RB}$
$(-172798217)_{10}$ $(01010100110100101000100011001000110011000101001110010110001010 10)_{RB}$

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.

```
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')                   {* Shifting phase }
           then east > pln0;
           else center > pln0;
not(center') > pln1;                        {* Rules for the other Planes }
& center > pln2;
& center' > pln2;
end.
make-table algebraic-sum
```

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

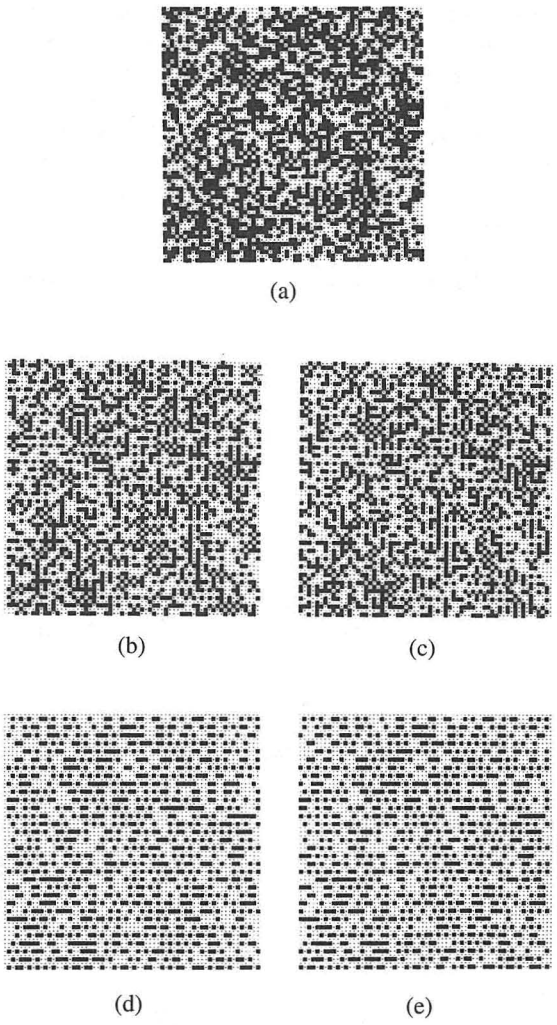

Figure 3: Execution of the local rule on a CA of size $N = 64$.

environment. The algebraic sum is executed in parallel in constant time by using the RB representation of numbers. The method presented in [5], which performs binary addition on CAs in linear running time, is enhanced.

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

[1] G. Bilardi and F. Preparata, "Horizons of Parallel Computation," *Proceedings of Symposium of* 25^{th} *Anniversary of* INRIA, Springer-Verlag LNCS, **653** (1992) 155.

[2] A. Califano, T. Toffoli, and N. Margolus, *CAM-6 User's Guide, version 2.1*, (1986).

[3] G. Jacopini and G. Sontacchi, "Reversible Parallel Computation: An Evolving Space Model," *Theoretical Computer Science*, **73** (1990) 1.

[4] T. Toffoli and N. Margolus, *Cellular Automata Machines—A New Environment for Modeling*, (Cambridge, MIT Press, 1987).

[5] B. Shet, P. Nag, and R. W. Hellwarth, "Binary Addition on Cellular Automata," *Complex Systems*, **5** (1991) 479.

[6] H. L. Garner, "The Residue Number System," *IRE Transaction on Electronic Computers*, **EC-8** (1959) 140–147.

[7] C. C. Guest and T. K. Gaylord, "Truth-table Look-up Optical Processing Utilizing Binary and Residue Arithmetic," *Applied Optics*, **19** (1980) 1201–1207.

[8] K.-H. Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution," *Applied Optics*, **25** (1986) 3054–3060.

[9] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Transaction on Electronic Computers*, **EC-10** (1961) 389–400.

[10] R. P. Bocker, B. L. Drake, M. E. Lasher, and T. B. Henderson, "Modified-signed Addition and Subtraction Using Optical Symbolic Substitution," *Applied Optics*, **25** (1986) 2456–2457.

[11] A. K. Cherri and M. A. Karim, "Modified-signed Digit Arithmetic Using an Efficient Symbolic Substitution Logic," *Applied Optics*, **27** (1988) 3824–3827.

[12] G. A. De Biase and A. Massini, "Redundant Binary Number Representation for an Inherently Parallel Arithmetic on Optical Computers," *Applied Optics*, **32** (1993) 659–664.

[13] G. A. De Biase and A. Massini, "High Efficiency Redundant Binary Number Representation for Parallel Arithmetic on Optical Computers," *Optics & Laser Technology—Special Issue on Optical Computing*, **26** (1994) 219–224.