Complex Systems **5** (1991) 479–486

Binary Addition on Cellular Automata

Bhavin Sheth
Prantik Nag
Robert W. Hellwarth
University of Southern California, Los Angeles, CA 90089-0484, USA

Abstract. A cellular automata approach to fast addition of binary numbers is presented. This approach utilizes the intrinsically parallel nature of the cellular automaton to add binary numbers represented as on/off pixels on the computer screen. Numbers n-bit long are added serially pairwise until the final sum is obtained. This approach can easily be generalized to numbers of any base, but is constrained by the number of planes available on the Cellular Automata Machine.

The concept of cellular automata was introduced by von Neumann and Ulam to model physical, chemical, and biological systems. Today cellular automata are used for several different purposes. One such application of cellular automata is in the area of computation. It has been shown that cellular automata can be used as general purpose computers and may therefore be used as general paradigms for parallel computation [3]. This paper concentrates on the capabilities of cellular automata for binary addition. The approach given in this paper to solving this problem is intended to emphasize the simplicity of cellular automata in implementing elementary mathematical operations. The operation of binary addition was implemented on a Cellular Automata Machine (CAM) designed by the Information Mechanics Group of the Massachusetts Institute of Technology. The rules discussed in the paper may assume a knowledge of the language constructs used (see [1]).

The approach given has certain unique features. The CAM screen may be filled up almost entirely with binary numbers, thus allowing as many as 256 numbers, each having a length of up to 256 bits, to be added. These limitations are solely machine dependent, since CAM only allows a screen of 256×256 pixels. It must be mentioned here that the approach does not take care of overflow. To understand this approach, it might help to visualize the numbers as beads on an abacus. The topmost pair of numbers are added first, and the sum obtained is then added to the number appearing beneath them. This goes on until all the numbers are exhausted, after which all that remains is the final sum of all the numbers. Henceforth the final sum is added with 0 until stopped. In adding two numbers, the corresponding

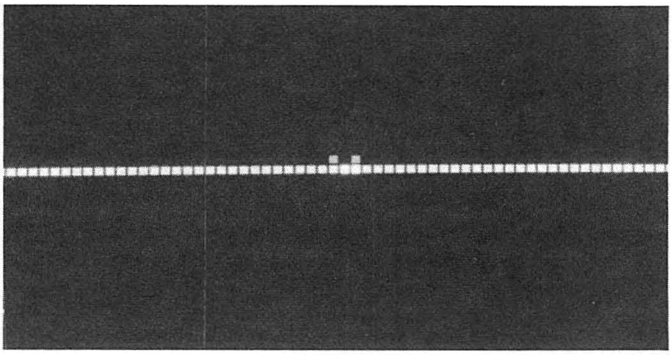


Figure 1: The numbers 101_2 and 011_2 are to be added. The row of ones on plane 2 can be seen clearly. The number 001_2 lies in the row right below the number 101_2 and on the same row as the ones on plane 2.

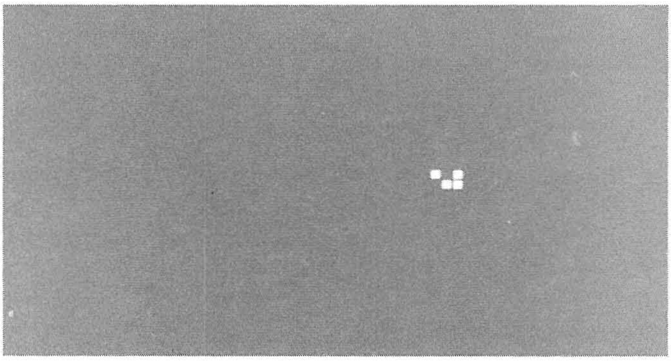


Figure 2: The numbers are shown. Plane 2 is hidden by an appropriate color map. (See below for the color map.)

bits are added in parallel and the carries are obtained. These carries are added to the partial sum obtained in the next step. Once again carries will be obtained, and will be added to the next higher bits in the following step. This procedure repeats 255 times, by which time the sum is obtained.

A binary number is represented as a series of on/off pixels on the display. A binary '1' will appear as a one on plane 0 and zeros on the other three planes. A binary '0' will be a zero on all four planes. A binary number is represented length-wise on the screen, with the left-most bit being the most significant bit.

Let us first examine the addition of two binary numbers.

The presence of the row of ones on plane 2 serves to distinguish between the two numbers. After the completion of the first step, the number at the top disappears. This disappearance of the number is taken care of by

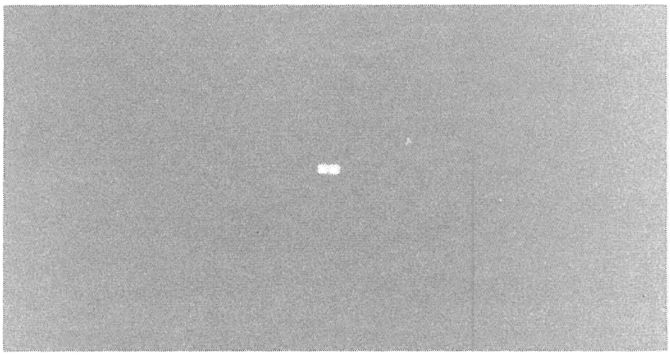


Figure 3: The first step is executed. The corresponding bits are added and the carries are saved and will be added to the number in the following steps. Plane 0 at present has 0110, while plane 1 has 0010, which will be added to the number 0110. Note that the carries will be shifted to the right by one bit, but will be added to the appropriate bits by the use of EAST.

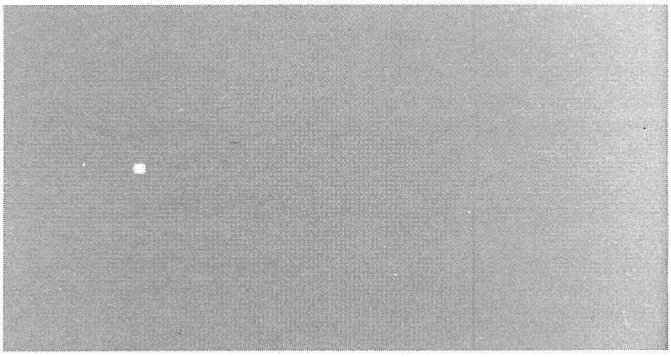


Figure 4: The carries are added. Plane 0 now has 0100, while plane 1 now has 0100.

the statement NORTH CENTER XOR &CENTER AND >PLNO. The row of ones on plane 2 disappears at the end of the second step, its purpose served.

The rule is shown below.

```
NEW-EXPERIMENT
N/VONN &/CENTERS


                                         : ADD_R1&R2

NORTH CENTER XOR &CENTER AND >PLN0
            NORTH CENTER AND >PLN1
                      CENTER >PLN2;
```

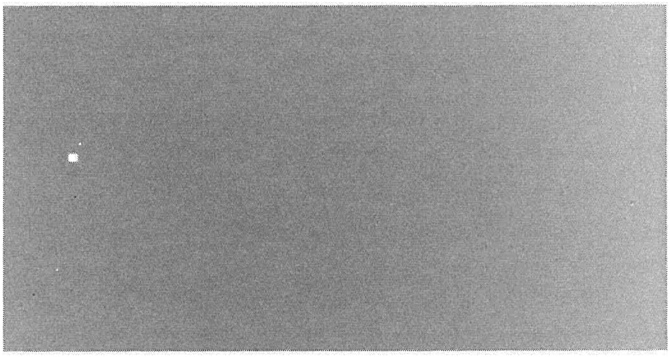


Figure 5: The final result, 1000_2. The number will be on plane 0. Note that the carries too may be hidden if desired. Also note that the planes have been shifted to the middle of the screen for clarity. In any case the answer would still have been correct.

```
MAKE-TABLE ADD_R1&R2

                          : CARRY-PROPAGATE
CENTER EAST' XOR >AUX0
CENTER EAST' AND >AUX1 ;
MAKE-TABLE CARRY-PROPAGATE

                                   : BINCYCLE
REG-TABS STEP AUX-TABS 255 STEPS;
MAKE-CYCLE BINCYCLE
```

As can be seen, the rule has two parts. `Add R1 & R2`, which takes one step, adds the two binary numbers represented in the adjacent rows. The carries are neglected and are stored in plane 1 in the same position. These carries are later added by the `Carry Propagate` part. The `Carry Propagate` part adds the carry from bit $i-1$ to bit i, for $0 \leq i \leq n$. The same binary operation found by `Add R1& R2` is used. The final sum is retained even after the addition is complete, owing to the property of the XOR operator. To add more than two binary numbers, some modifications were made to the initial pattern file and to the rule.

First, the row of ones on plane 2 keeps moving down once the two binary numbers are added. Hence the statement

```
NORTH >PLN2
```

Thus, in intermediate steps, the row has the partial sum of all the numbers above it. At the end of the final step, it possesses the final sum of all the numbers, as in the rule that added two binary numbers.

Note that all the numbers above this row that stores the intermediate sum—this row has the ones on plane 2—should be deleted. However, the

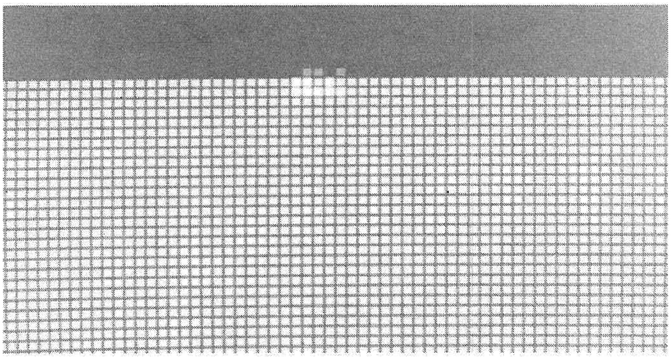


Figure 6: The bright pixels at the bottom portion of the screen are all ones on plane 3. The numbers to be added are 01101_2, 11010_2, and 11111_2. The uppermost row of ones on plane 3 also happen to have ones on plane 2.

numbers below this row should be maintained since they are still to be added. The initial pattern file is changed somewhat to accommodate this fact. The part of the screen below the row that has the sum has ones on plane 3. This is to distinguish the numbers that have not been added from the numbers that have already been added. In the beginning, only one number lies above this row. Just as the row that keeps the intermediate sum keeps going down, so must the block of ones on plane 3. This is taken care of by the statement

```
    NORTH >PLN3
```

This spatial differentiation between the numbers is used in

```
CENTER &CENTER' &CENTER NOT AND AND
        NORTH CENTER XOR &CENTER AND OR >PLN0
```

and in

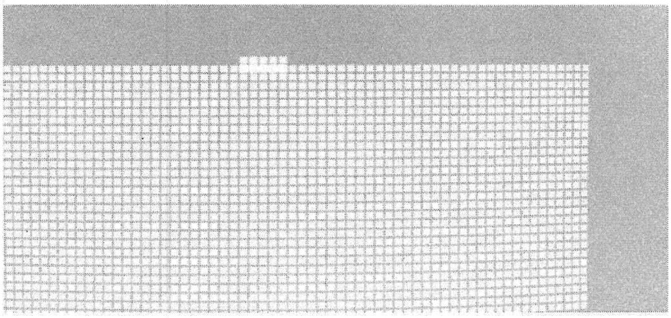
```
CENTER' CENTER AND &CENTER' NOT CENTER EAST
                               XOR AND OR >AUX0
```

The rule, in its entirety, is given below.

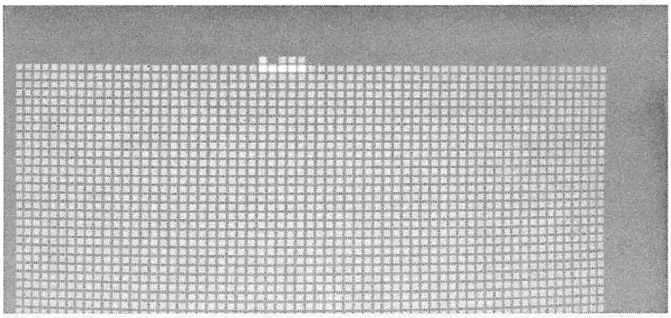
```
NEW-EXPERIMENT
N/VONN &/CENTERS

                                                    : ADD_R1&R2
CENTER &CENTER' &CENTER NOT AND AND
           NORTH CENTER XOR &CENTER AND OR >PLN0
              NORTH CENTER &CENTER AND AND >PLN1
                                     NORTH >PLN2
                                    NORTH' >PLN3 ;
MAKE-TABLE ADD_R1&R2
```

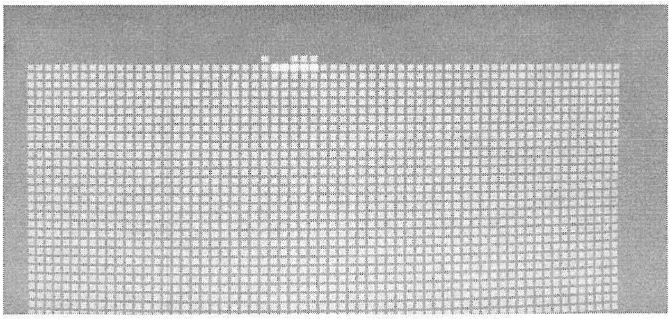
(*a*)



(*b*)



(*c*)



(*d*)

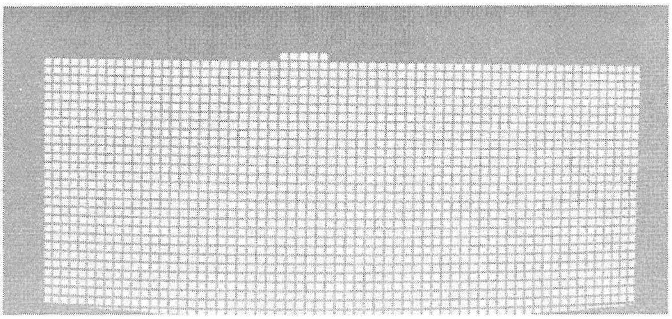
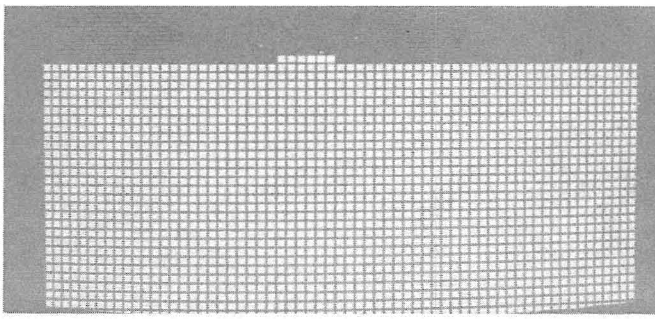
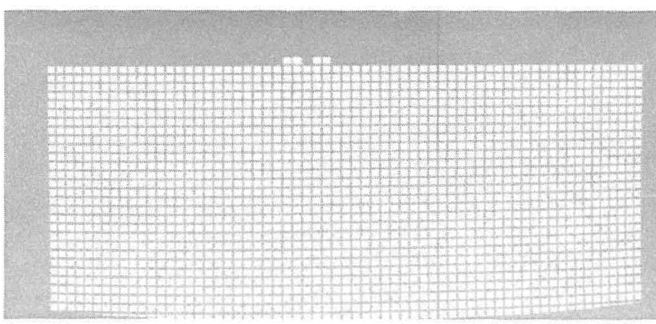


Figure 7: Some of the intermediate steps.

(*e*)



(*f*)



(*g*)

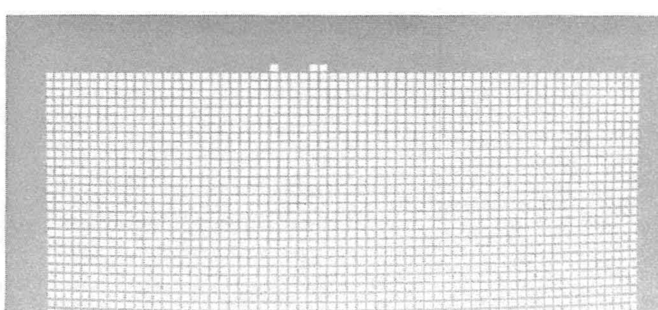


Figure 7: *Continued.*

```
                                        : CARRY-PROPAGATE
&CENTER' CENTER AND &CENTER' NOT
                  CENTER EAST' XOR AND OR >AUX0
       &CENTER' NOT CENTER EAST' AND AND >AUX1
                                   CENTER >AUX2
                                  CENTER' >AUX3 ;
MAKE-TABLE CARRY-PROPAGATE

                                        : BINCYCLE
```

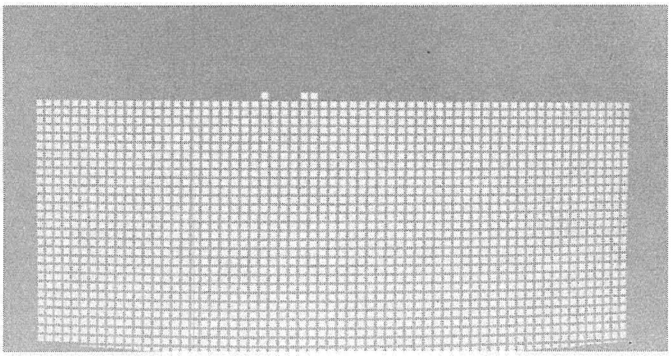


Figure 8: The final sum, 1000110_2.

```
REG-TABS STEP AUX-TABS 255 STEPS ;
MAKE-CYCLE BINCYCLE
```

This rule adds several binary numbers. The numbers may be thought of as partial products obtained from multiplication.

The rows of ones on planes 1 and 3 can be made invisible by the use of an appropriate color map. The following color map may be used:

```
           ALPHA >GREEN
          ALPHA' >BLUE
ALPHA ALPHA' AND >RED
               0 >INTEN ;
```

Conclusion

The experiment demonstrated that binary addition can be modeled as a cellular automaton rule.

Acknowledgments

This research was supported by the U.S. Air Force Office of Scientific Research under contract F49620-91-C-0018, and by the National Science Foundation under grant ECS-8821507.

References

[1] T. Toffoli and N. Margolus, *Cellular Automata Machines* (Cambridge, MIT Press, 1987).

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

[3] Stephen Wolfram, *Reviews of Modern Physics*, **55** (1983) 601–644.