MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence
Memo No. 198

June 1970
First Draft

CELLULAR AUTOMATA

Edwin Roger Banks

This paper presents in order 1) a brief description of the results, 2) a definition of cellular automata, 3) discussion of previous work in this area by Von Neumann and Codd, and 4) details of how the prescribed behaviors are achieved (with computer simulations included in the appendices). The results include showing that a two state cell with five neighbors is sufficient for universality.

Work reported herein was supported by the Warren McCulloch Laboratory, an M.I.T. research program sponsored by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-70-A-0362-0002.

Reproduction of this document, in whole or in part, is permitted for any purpose of the United States Government.

CELLULAR AUTOMATA

INTRODUCTION

Complex behavior by machines can be achieved by either having a large number of very simple machines or by having a complex machine with which to start. Our primary interest in this paper is with the former. By considering the global behavior of a large number of the simplest of machines, the following results are shown:

1. An array of identical square cells each of which has only four states and communicates with its four nearest neighbors (forming a neighborhood of five cells) can a) perform any computation which is computable and b) construct (almost) any configuration --in particular, it can be self-reproducing. Cells capable of the first behavior are called universal computers; the second behavior characterizes the universal constructor.
2. A three state, five neighbor cell is capable of universal computation when configured in a finite initial area.
3. Two states and five neighbors are sufficient for universal computation, but require an infinite initial configuration.

Being parallel machines, these cellular automata can serve as a good theoretical basis for parallel computation and should be useful mathematically in many of the same areas as the Turing Machine. Practical physical applications are presented in a later section.

CELLULAR AUTOMATA

Consider an infinite array (in two dimensions) of identical simple finite state machines, which we shall call cells. Each cell communicates with other cells in its neighborhood. The finite state machines, or cells, are described by specifying a set of transition rules. These rules specify which state a cell will enter (during the next time step) as a function of the cell's neighbors. Except as otherwise specified, we will consider only two dimensional square cells with the four nearest neighbors comprising the five-neighbor neighborhood.

A configuration is a specification of the states of the cells in the array at any given time. When started in particular initial configurations, the cells can be made to perform interesting processes.

In summary, we are considering the possible behaviors exhibited by vast numbers (usually infinite) of identical, very simple machines each interacting with its nearest neighbors.

A particular cellular space is characterized by the transition rules. We will use the form

(CNESWR)

to write the rules. The letters stand for Current state, state of the North neighbor, East, South, West, and the Result state R. A set of rules in this form defines a cellular space.

Several conventions must be observed. First, only _transition_ rules will be listed in the set. If a neighborhood configuration does not appear (implicitly or explicitly -- see below) in the set, then it is understood that the cell does not change state in the next time period. Second, there are varying degrees of symmetry. For example, if there is no preferred direction, then the appearance of any of the following rules in the set implies the implicit presence of the others.

(cabbbd) (cbabbd) (cbbabd) (cbbbad)

Also if there is no preferred rotation (clockwise or counter-clockwise), then the following are equivalent and only one needs to appear in the rule set.

(cabdde) (cbadde)

All the sets of transition rules in this memo have the above two properties -- no preferred direction nor rotation.

One other requirement is always observed -- the existence of a _quiescent_ _state_. A quiescent state is a state that remains in the quiescent state when all its neighbors are quiescent. Thus another characterization of cellular configurations is the number of non-quiescent states required for the initial configuration.

PREVIOUS WORK

J. Von Neumann pioneered in the area. His reference (1) is to be taken as the foundation, but the book by Codd (2) is the chief reference for this work. Von Neumann's primary interest was in finding a set of rules and initial configuration that would be capable of self-reproduction, in particular, and universal construction in general. The configuration was also required to be a universal computer. The configuration functioned by "growing" an arm which could "construct" a new passive configuration. An activation signal sent along the arm started the new construction operating. His solution was the twenty-nine state, five neighbor cells (four nearest neighbors plus itself). The set of rules were not isotropic. E. F. Codd (2) reduced the required number of states to only eight. His transition rules specified an isotropic space, i.e. no preferred direction, but the rules did possess a preferred rotation. Both Von Neumann and Codd worked with the following requirements.

1. The cellular machines are initially configured in a finite region of space -- i.e. only a finite number of non-quiescent states existed in the initial configuration.
2. Construction was performed in an initially quiescent region of space.

Codd (2) has also shown that two states are sufficient for universality if the neighborhood is allowed to be increased. In particular, he has shown that a two state cell with eighty-five neighbors can simulate his eight state cell.

Considering only self-reproducing patterns, Edward Fredkin has described the following interesting cellular space. If the states are "0" and "1" and if the result state is obtained by taking the sum of the neighbors, modulo 2, then any initial configuration will reproduce copies of itself symmetrically about the initial configuration. Terry Winograd generalized this result showing that any neighborhood, not necessarily just the four nearest neighbors, and any number of dimensions still give the same results. Further, if there are p states 0, 1, ... , p-1 where p is a prime number, then the sum of the neighbors modulo p is a rule that will assure self-replication of the pattern in a finite number of time steps.

DETAILS OF THE CELLULAR AUTOMATA

I. The Two State Universal Cellular Space

One of several methods of achieving universality with two states is to show that two states in the proper configuration can simulate any n-state cell -- in particular the twenty-nine state Von Neumann or eight state Codd cells. This is done by showing that cells in the proper configuration can represent wires and that signals can travel along the wires. Other regions are configured to act as junctions, crossovers, logic elements, curves, etc. Further elaboration will be given after it has been shown that these elements can be constructed.

If the two states are represented by "0" and "1", the three transition rules can be written.

(111000) (011101) (011111)

Rules for the Two State Universal Computation Cellular Space

The first rule requires corners to disappear. The other two assure that gaps (zeros) surrounded by three or four ones will be filled in. In the following illustration of a signal propagating along the wire, the "0" or quiescent state is also denoted by the blank.

```
11111111111110111111111111111
11111111111101111111111111111
11111111111111111111111111111
```

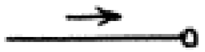
Wire and Signal (Propagates to the Right)

Note that the signal travels on one side of the wire. The wire will be symbolized by a straight line with an arrow used to indicate the side of the wire on which the signal is traveling. It should be clearly understood that the blank area above and below the above diagram represents cells in the zero state and not the absence of cells. Appendix I contains computer simulations of the above wire and also the other elements shown below. The reader should probably check this Appendix now to see exactly how this signal propagates.

The dead-end is used to eliminate signals traveling down truncated wires.

```
              1
111111110111111
111111101111111
111111111111111
              1
```

or



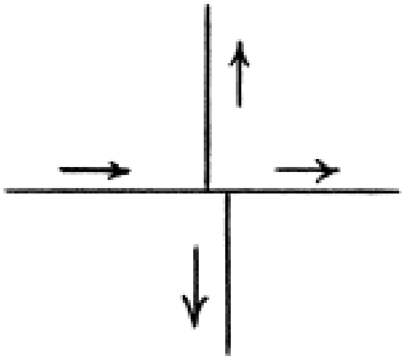
Symbol

The Dead-End

The junction or fan-out is used to create new signals. A signal entering the fan-out from the input side leaves from the other three arms. This element will also be used with the above dead-end to produce the curve.

```
               111
               111
               111
               111
111111111111111111
11111111111111111111111111111
111111111111111   11111111111
             11   11111111111
             11   111
            1111  111
                  111
                  111
```

or



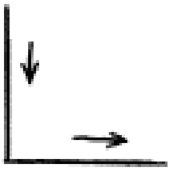
Junction (and Diode)

This fan-out is not bilateral -- a signal entering any of the side outputs dies. Thus by using two dead-ends the diode (or one-way gate) is obtained.

A whole family of curves must be created since the signal is on one side of the wire. We need inside to inside, inside to outside, outside to outside and outside to inside curves. Although simpler curves exist, two of these curves are obtained directly from the fan-out with dead-ends on two output wires. The inside to inside curve must be created from scratch.

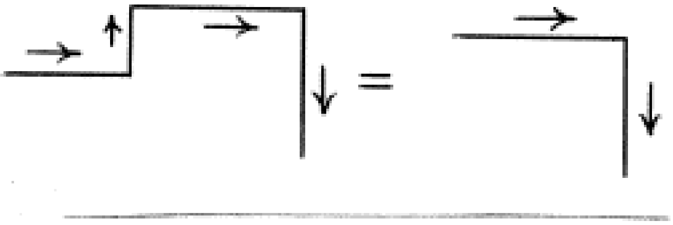
```
 111
 111
 111
1111
 111
 111
1111    1
  1111111
  11111111111111111111
  1      1111111111111
         1111111111111
         1
```

or



The Inside to Inside Curve

The outside to outside curve can be made from the above curves as follows.



The Outside to Outside Curve

Recall that the arrows indicate which side of the wire carries the signal.

A universal logic element will be used with a clock (described later) to build the remaining needed elements. The following configuration will compute the logic function "B and NOT A".

```
          111
          111
       A↓ 1111
   B      111
 ───→     111
1111111111111    Ā&B ───→
11111111111111111111111111111111
1111111111   1111111111111111111
        11   1111111111111111111
        11   1
       1111
```

TRUTH TABLE

B \ A	0	1
0	0	0
1	1	0

Logic Element

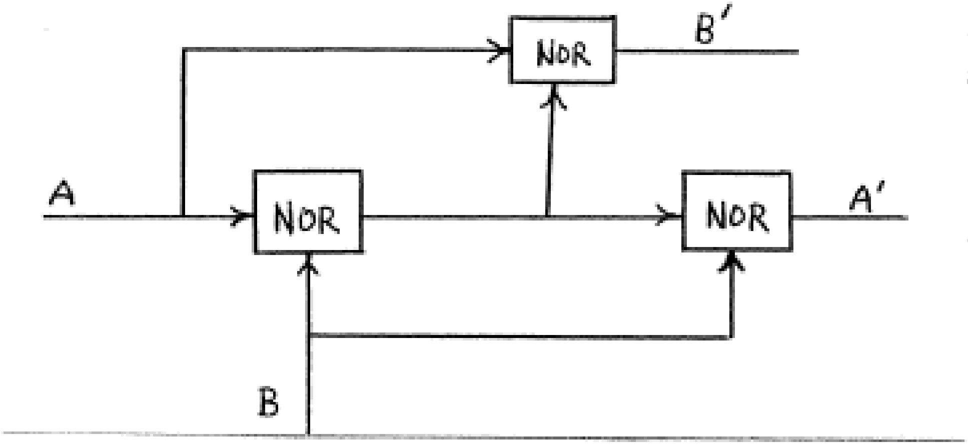
If the B input is from a clock (i.e. a periodic emitter of signals) then this logic element becomes a "NOT" function. Thus we need a clock.

```
         1
         1111111111111111
 1       0111111111111111
 111111111011111111111111
111111101
        1
```

The Clock

This clock has a period of sixteen time units. Almost any even period is obtainable by different configurations.

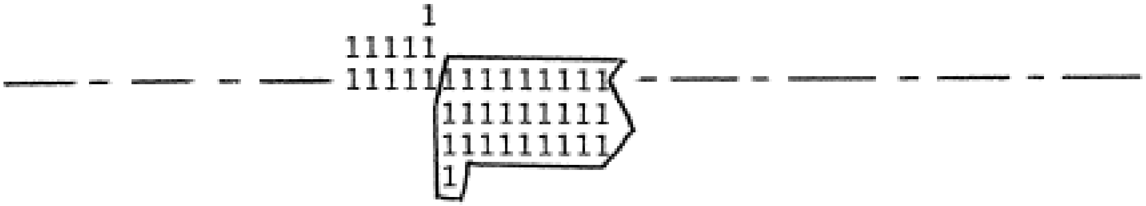
The operation of the logic element requires that the signals be synchronized. The clock defines an interaction time and all wires are constructed to maintain the synchronization of signals. The construction of the crossover is achieved by the following logical configuration. However it is required that only one signal arrive at once. By moving the location of the crossover slightly, non-simultaneous arrivals can be assured. The NOR function can be constructed from the logic element and the NOT function.



The Crossover

A much more efficient crossover is illustrated in Appendix II.

The extension to three dimensions and seven neighbors is trivial. The same three rules apply assuming the other two neighbors are in the "0" state (previously there were only five neighbors.) The above machine can exist in a plane, if desired, but turns out of the plane can be inserted at any point of origin of signals in the wire.



Part of the Inside to Inside Curve

The enclosed part of this curve can be rotated ninety degrees about the dashed line.

These elements are sufficient for universality. The wire and crossover allow moving of signals from point to point. The logic function with the NOT function added to its "B" input gives the universal NAND function. (See Minsky (3) for a discussion of the universality of this logic function.) Thus we can "wire up" any function. (Delays, which might be required, can be accomplished with a few extra curves.) Specifically, the twenty-nine state Von Neumann cells or the eight state Codd cells could be simulated achieving both computation and construction universality except for the fact that an infinite number of simulated cells must exist.

As an alternate approach to achieving computation universality, the Turing Machine can be simulated. We wire up an infinite Turing Machine tape with the finite state machine part built into each tape cell. An activation signal causes only one of these simulated tape cells to be operative at once, with both state and activation signal passed to the left or right. (This approach will be used later in showing the universality of a one dimensional cellular automaton.)

DETAILS OF THE CELLULAR AUTOMATA

II. The Three State Universal Finite Cellular Space

The undesirable requirement of an initial configuration involving an infinite number of cells in the above two state cellular space exists. This requirement can be eliminated by adding another state. (Codd (2) conjectures that an unbounded but boundable propagation is a necessary condition for computation universality and uses this conjecture in a proof that there does not exist a two state five neighbor universal cellular space with finite initial configuration.)

Since a computation may require an arbitrary amount of space some method must exist for increasing the information storage by arbitrarily large amounts. The method is to use an arbitrarily extensible special wire with the property that a signal sent out this wire will lengthen the wire by a constant amount with a reflection or echo signal returned back down the wire. Four of these special wires will allow simulation of Minsky's universal two register machine (3). A brief description of the operation of this machine is necessary.

The two register machine operates on two infinite capacity registers. This is a program machine with the operations a) subtract 1 from a register and if the result is zero, branch to a specified operation, and b) add 1 to a register. Since addition and subtraction are operations that require only a finite state machine (as opposed to multiplication), only the ability to add and subtract 1 from the registers and to test for zero in the registers is needed. Two of the special extensible wires are used for each register. The difference in length represents the number contained in the register. To add 1, a signal is sent down one of these wires extending it. The echo is ignored or destroyed by meeting another signal sent down the wire before the echo returns. (Two meeting signals will be anihilated.) The zero test is achieved by comparing two echo signals for simultaneous return and subtraction is done by lengthening the shorter wire. To prevent "almost simultaneous" returns, the extending of wires can be done several times so that the length changes only by increments of sufficient amount.

Let's look at the needed components. Appendix III contains the computer simulations of these elements and a listing of the transition rules. The states are "2", "1", and the quiescent state "0" (also represented by blank space).

The wire is composed of 2's with the signal represented by a one-zero train.

```
        →
222222222222222222222222222
2222222222 1222222222222222
222222222222222222222222222
```

Wire (Propagation to the Right)

The following junction construction

```
   222
   222
   222
222222222
222222222
222222222
   222
   222
   222
```

The Junction

has the properties 1) a single signal entering an arm exits from the other three arms, 2) two signals entering simultaneously at right angles will exit the other two arms, and 3) three signals entering simultaneously are anihilated. In each case the junction is restored to its previous condition before arrival of a signal or signals.

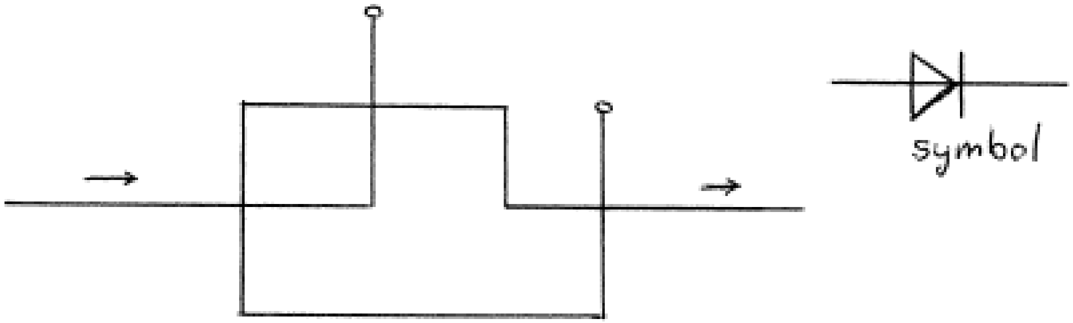
The dead end is simply a chopped off wire. However, if an extra "2" is placed at the end of the dead end wire as shown

```
 22222222222
222222222222
 22222222222
```

Special Wire

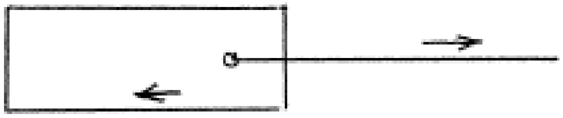
then the signal will reflect with the special wire being lengthened by two cell edge lengths.

We now illustrate that the above components form a universal set of elements. The curve is obtained from a junction with two dead ends. Another element needed is the diode (one-way gate). Representing wires by lines, dead-ends by small circles and curves as right angles we have



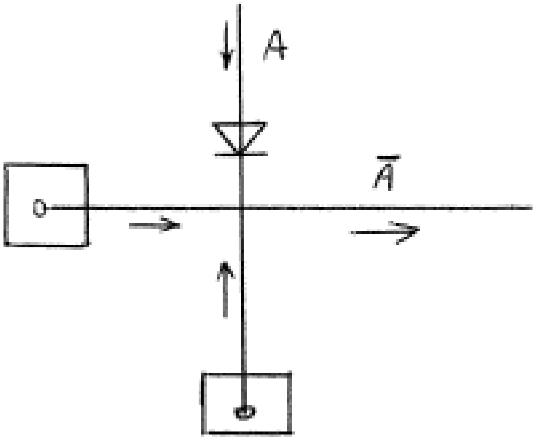
The Diode (The Signal will Pass only from Left to Right)

The clock is simply a signal circling in a loop with an exit.



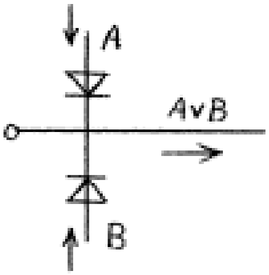
The Clock

The diode and clock are used to make a NOT function



The NOT Function

The OR function is just a junction with diodes.



The OR Function

The NOR function is obtained from the NOT and the OR. The NOR can be used as in the two state case to form a crossover.

The reception of echo signals, the program logic, and comparison of return times are now achieved by straight forward constructions. Thus the finitely initial configured universal three state cellular automaton can be built.

DETAILS OF THE CELLULAR AUTOMATA

III. The Four State Universal Computer Constructor

The four state universal computer, universal constructor is considerably more complex than the previous two automata. Not only must this automaton be capable of computing any (computable) computation, but it must be capable of constructing into quiescent, empty space an automaton also capable of computing any computable function. In particular, it should be capable of reproducing itself. (For a discussion of how such offspring can evolve and improve, see von Neumann(1).)

This automaton will have the same type logic elements as the previous machines, but in addition, it must have an arm that can reach out into the construction space to build the new machine. The growth and operation of this arm is quite complicated. After the new machine is constructed, it must be activated. (The approach of constructing an active machine directly would be tremendously more complicated.) The explanation of how an unbending arm can construct a new machine of larger size than the constructing machine is then explained.

Simulations of the components and a listing of the transition rules is given in Appendix IV. The states are represented by "0" (and blank), "1", "2", and "x".

The place to begin is with the wire and signal, and the dead-end.

```
       ←
xxxxxxxxxxxxxxxxxxxxxxxxx
x      12
xxxxxxxxxxxxxxxxxxxxxxxxx
```

The Wire and Signal and the Dead-End

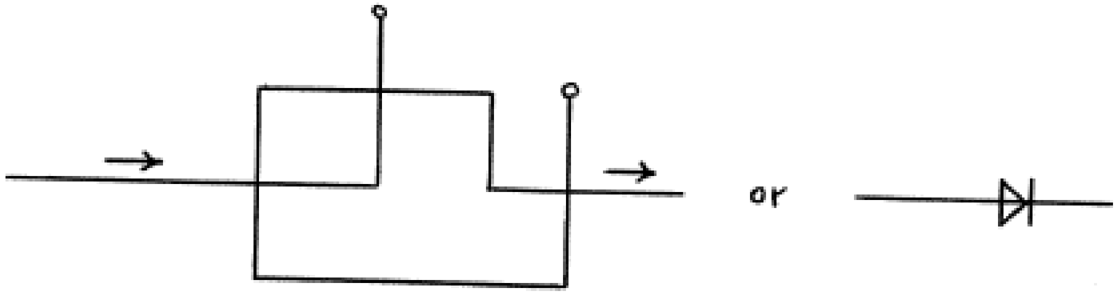
The wire and all other components are constructed from state "x". In fact, any configuration constructed entirely from this state will be passive. The junction

```
  x x
  x x
xxx xxx
   x
xxx xxx
  x x
  x x
```

The Junction

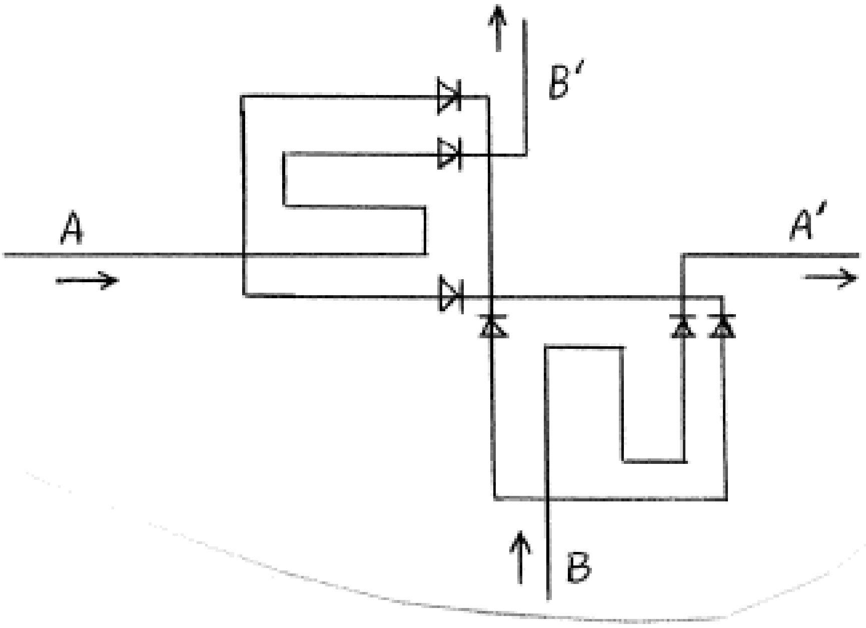
has the same properties as did the three state automaton, i.e. a) a single entering signal fans out, b) two inputs at a right angle give two outputs on the other two wires, and c) three input signals are anihilated. In each case the junction is restored to its original position. In the case of the fanout, the useful phenomenon of an extra delay of one time unit occurs.

The curve is obtained from the junction with dead-ends. The diode is obtained as previously.



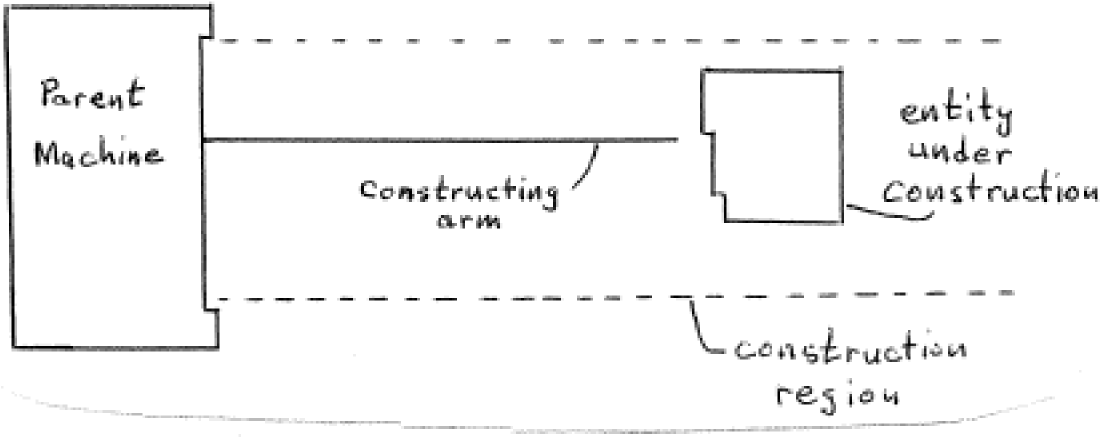
The Diode

Similarly the clock, NOT function and NOR function can be constructed. However the previous crossover is unacceptable because it involved the use of the NOR function which is built with a clock. The reason for this unacceptability is that we only want to construct a passive configuration and to use a single activation signal to introduce a signal into all wires and clocks to "start" the machine. With a passive crossover, there is no problem. It seems likely, however, that only the active crossover clocks associated with the distribution of the activation signal could be initialized during construction. Nevertheless the following passive crossover eliminates the problem.



The Passive Crossover

The logic operations of this automaton are completed. The operation of the arm is now illustrated.


Parent Machine
constructing arm
entity under construction
construction region

Construction

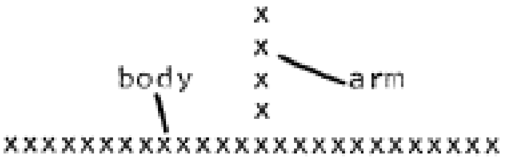
The horizontal arm can move vertically within a fixed range, but can extend an arbitrary distance into a preliminary construction region. The construction of automata which do not fit within this space (e.g. self-reproducing) is achieved by constructing a preliminary automaton of arbitrary length which grows a vertical arm into the surrounding space to construct the desired automaton.

The arm consists of a row of cells in state "x".

```
xxxxxxxxxxxxxxxxxxxxxxx
```

The Arm

The arm is attached to a row of cells in state "x".



Attachment of Arm to Body

The operation of construction proceeds as follows. 1)Two special signals (called single wings) sent from each end of the body collide at a location determined by the timing of signal origination. 2) The collision produces an arm bud. 3) Further pairs of single wings colliding at this location cause the arm bud to grow longer. 4) Once the arm has reached a length of three x's further growth is accompanied by an echo signal from the end of the arm. When the echo returns to the body, it is harmlessly anihilated unless, 5) if the echo signal is properly timed with another pair of colliding single wings, a new signal (called the erasing double wing) propagates out the wire, destroying the wire as it goes, i.e. leaving quiescent cells behind it. 6) When the erasing wing reaches the end of the arm, a single "x" is deposited into the cell just beyond the end of where the arm was. This depositing succeeds even if there are other x's surrounding the new "x" on one or two sides. 7) A new arm is grown to deposit other x's and the process is iterated until the construction is completed. Then an activation process takes place.

The above operation description requires some new components. First the single wing is illustrated.

```
         ──→
        1
xxxxxxxxx1xxxxxxx
```

Single Wing Signal

Two meeting single wings leave the following arm bud.

```
         x
         x

xxxxxxxxxxxxxxxxxxx
```

Arm Bud

Another collision fills in the gap in the arm bud forming an arm of length three. Another collision gives the following configuration.

```
         x
         x
         x

xxxxxxxxxxxxxxxxxxx
```

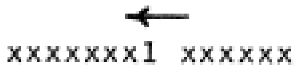
Arm with Gap

Again the gap is filled by another collision. Subsequent collisions give the following double wing signal propagating out the arm

```
         x
         x  ↑
         1  |
        1x1 |
         x
xxxxxxxxxxxxxxxxxxx
```

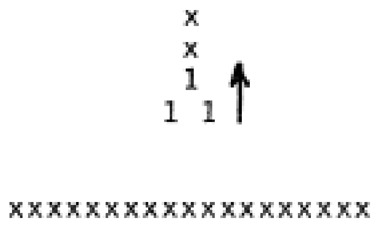
Double Wing Signal

This double wing adds an "x" to the end of the wire and sends the following echo signal back down the wire.



Echo Signal

This echo leaves a gap between the arm and the body which is filled in as before. However if the echo coincides with another collision of single wings, the erasing wing is produced.



The Erasing Wing Signal

The different cases of the erasing wing reaching the end of the wire are illustrated in the simulations. By constructing the new automaton column by column and top to bottom in a kind of "raster scan" the arm will need to consider only local configurations in which there are no x's already there or one "x" on the previous column or beside it in the current column or there may be two x's already there in these two positions. In particular, the arm will not have to fill an "x" between two existing x's.

The generation of the single wing from the original logic signal is achieved by the following element.

```
       x
xxxxxxxxx   1
  21     xxxx1xxxxxxxxxxx
xxxxxxxxx   --->
 --->   x
```

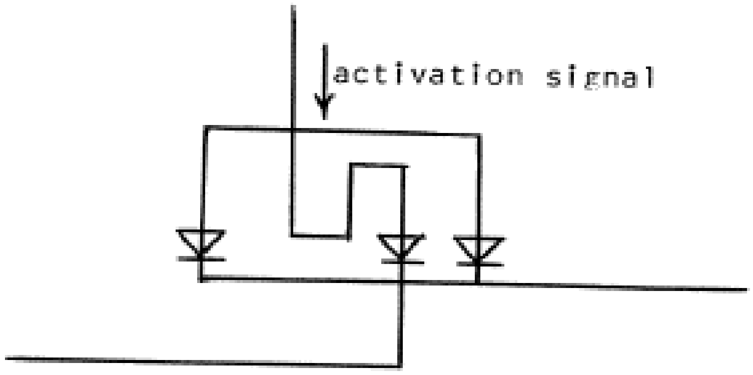
Single Wings Being Generated

Activation is achieved by the capture mechanism.

```
xx
xxx
  xxxxxxxxxxxxxxxxxx

  xxxxxxxxxxxxxxxxxx
xxx
xx
```

Capture Mechanism

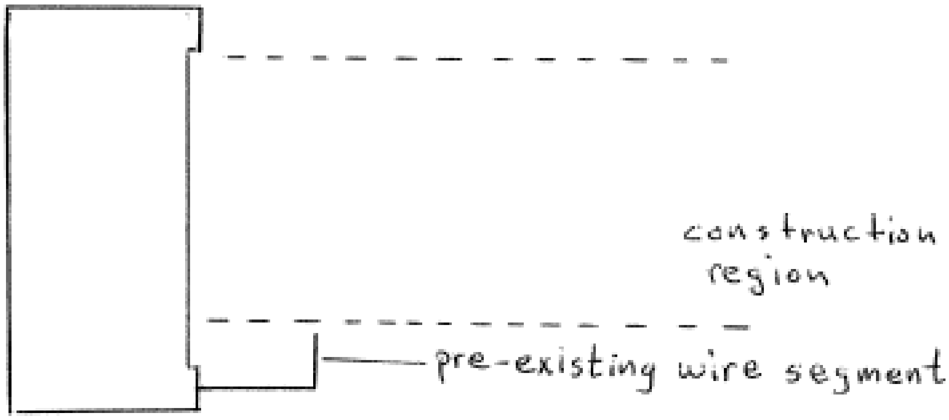
An erasing wing entering the above mechanism will generate a "12" signal and seal the end forming a dead-end. This activation signal travels by fan-outs and crossovers to every place where it is desired to have a wire or clock initialized with a signal. The following simple configuration allows the introduction of initial signals. Note that the wire maintains a two-way propagation capability.



Introduction of Right Moving Signal

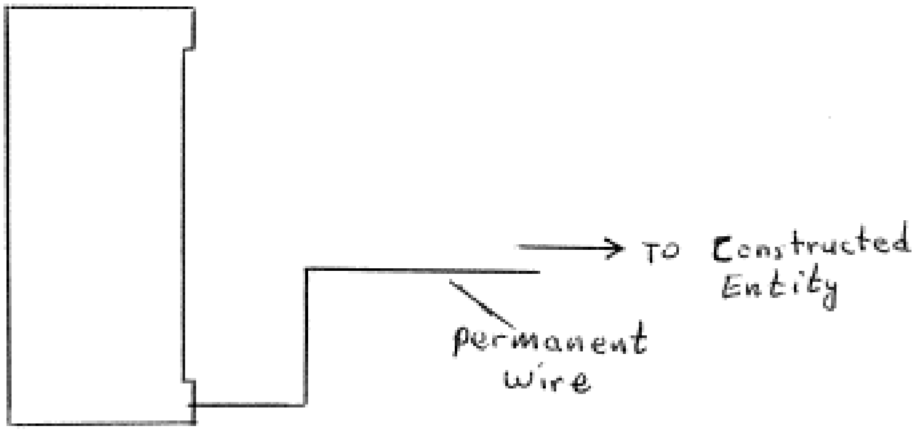
If some initial signals are desired to be within junctions or elements, they could have been introduced earlier into the input wires.

By suitable delay lines built into the activation wires, any finite amount of information can be communicated to the new machine at the expense of building a more complex machine. Other methods of communication involve simultaneous counters, one in each machine, being interrupted by a second activation-type signal (requiring a second capture mechanism) with the current count representing the information. If two-way communication is desired, the following technique may be used.


construction region
pre-existing wire segment

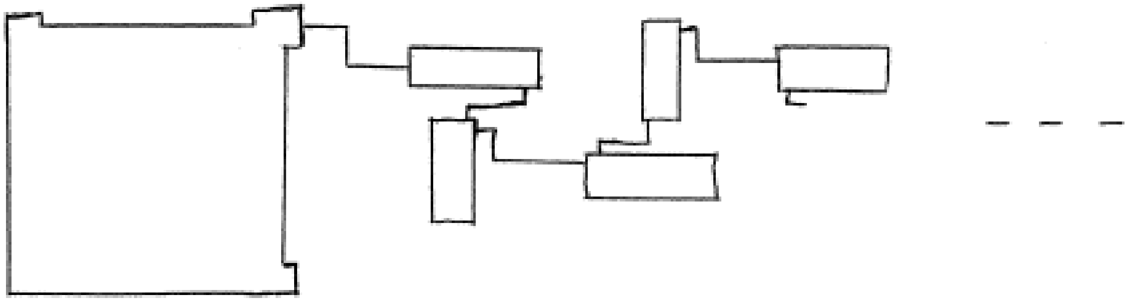
Starting Configuration

The constructed automaton, instead of being activated through a capture mechanism, is simply linked to the pre-existing wire segment.


TO Constructed Entity
permanent wire

Final Configuration

To function as a universal computer, the machine needs an infinitely extensible memory, or information storage capacity. This memory is achieved by having a separate construction arm that constructs a new memory box with its own logic, (finite) memory, and construction arm to extend itself when its memory capacity has been exceeded. The memory box is connected by a two-way arm (as illustrated above) to the parent automaton and to the next memory box in the series. The series of memory boxes can be thought of as a memory tape.



Memory Boxes

Consideration of self-reproducing automata yields the following new problem. When constructing offspring, there are two configurations in which this offspring could be desired. The first is that the offspring should be constructed in some original configuration of the parent. This is achieved easily and straight-forwardly. The second, however, would have the memory reached zero, a third activation type signal sent to the offspring would stop its counting and cause both offspring and parent to start moving the reserve memory (now identical) back into the main memory. Of course, suitable delays would be needed to assure proper synchronization between the final configurations.

offspring in some configuration after the parent has done some computation. In this case, the parent will have grown an arbitrary number of memory boxes.

To have the offspring have the same number of memory boxes as its parent, it should be constructed in such a manner that when activated, it immediately would begin to grow its own memory boxes or tape. This process would continue until it reached the same length as the parent's tape. It would know when to stop by a second activation type signal from the parent. (Note that permanent connection by a wire between parent and offspring has not been allowed.)

It must be noted that the memory boxes cannot be a unary storage as were the extensible tapes of the three state machine. Such a tape can store no more information than its own length, but here the parent's tape must contain information about its own length in order to know when to send the second activation signal to stop the offspring's tape's growth at the same length.

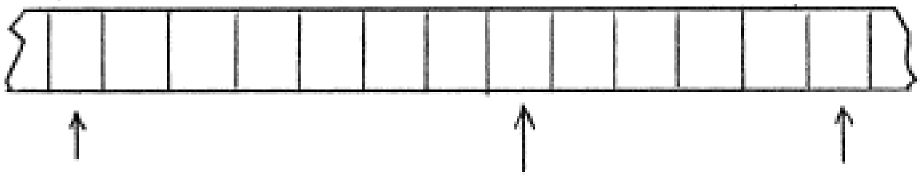
If information is desired to be in the offspring's tape, this can be achieved by having all memory boxes built with a duplicate reserve capacity. The offspring, after having stopped growing its memory, could begin counting in its reserve memory. Simultaneously, the parent would begin counting in its reserve memory and counting down in its main memory. When the main

DETAILS OF THE CELLULAR AUTOMATA

IV. Universal One Dimensional Cellular Spaces

The two state universal cellular automaton previously discussed can be constructed in a finite width. By considering an infinite one dimensional tape made up of slices of the two state machine configured in two dimensions, it is at once a universal computer with two states and five neighbors if two of the neighbors of a cell are its nearest neighbors and two others are some distance away, namely the width of the slices. The fifth neighbor is the cell itself.

An attempt was made to find a universal one dimensional cellular space with just the two nearest neighbors forming a three neighbor space. A seventeen state cell was found. This space achieves universality by simulating the above five neighbor one dimensional cell. If the simulated cells of each of the slices are distributed in a large region of the tape in a way to be discussed with



Location of Simulated Cells

quiescent space between them, then each cell could begin simultaneously to send out signals in both directions. There would be two types of signals depending on the initial state of a cell. The first two signals to pass over a cell would perform a partial computation of the resulting state, leaving it in one of five states. All subsequent signals passing over it would leave it unchanged unless the two signals arrived simultaneously. The only case in which two signals can meet simultaneously over a simulated cell is when the signals originated from the cell's two distant simulated neighbors. (No rigid proof has been found, or even intensively sought, that a distribution that meets this requirement exists, but it appears obvious that with a small enough density of simulated cells, some such distribution would likely exist.) This collision over the simulated cell would complete the computation of its transition and the process would start over. It should be mentioned that all other collisions have no effect--signals pass through each other unaffected, etc. Complete details will appear in a forthcoming thesis.

DISCUSSION

I. Starting and Halting of Cellular Automaton Computations

The starting and halting of Turing machine computations are well defined operations. Similar operations must be described for the universal cellular automata described.

In the same sense that the input to a universal Turing machine is initially configured on the tape, representing the computation to be executed, the initial configuration of a cellular automaton contains the description of the computation.

Halting of a cellular automaton can be defined in several ways. Probably the simplest is to define a computation as having ended when a specified cell first changes state.

DISCUSSION

II. The Equivalent Continuous Cellular Space

If we consider what happens to the two state cellular space as the size of the cells and the cycle time approach zero, we obtain a continuous "cellular" space. If the cycle time and size approach zero in such a way that the quotient of size divided by cycle time remains constant, a characteristic velocity is defined as this constant for the continuous space. Let the state (0 or 1 for our two state automaton) of the discrete cells correspond to the value of a function in the continuous space. Spatial derivatives can be defined. A particular set of transition rules can now be written in differential equation form using the derivatives.

The transition rule set for the universal two state cellular automaton has the interesting property that the corresponding differential equation can be written with the time derivative of the space function (state), $\dot{S}(x,y,t)$, in terms of $S(x,y,t)$ and only the two second derivatives of $s(x,y,t)$ with respect to the x and y directions.

Konrad Zuse (4) has presented an argument for a discrete model of physics. Paul Klein of Project MAC has also interested himself in a cellular model of the universe. Not limiting himself to two or three states, he has found configurations which can propagate through a space in an arbitrary direction and at an arbitrary speed.

DISCUSSION

III. Practical Application

Perhaps the predominant application of cellular automata theory will be in circuits with a self wiring capability. A large scale integrated circuit forming a large array of cells could contain an initial configuration with construction power. When activated, it would proceed to wire up any desired circuit, (as indicated by information built into the configuration or sent into it after activation) detecting and ignoring bad cells. Such a circuit could actually be a complete parallel or serial digital computer on one slice. Various regions of the slice could have different transition rules if, for example, one type cell proved more efficient for memory and another for logic functions. Variations include having no initial configuration wired in, but sending inputs in to build a constructing mechanism.

The main point is that by allowing bad areas to appear on the slice without destroying the total slice, much cheaper and more complex circuitry could be used. Shoup (4) has considered the design of integrated circuits used as cellular arrays. Not concerning himself with universality in particular, he was more interested in efficient implementations of practical operations.

REFERENCES

1. Von Neumann, John, Theory of Self-Reproducing Automata, Ed. by Arthur W. Burks, Univ. of Illinois Press, Urbana and London, 1966.
2. Codd, E. F., Cellular Automata, Academic Press, Inc., New York and London, 1968.
3. Minsky, Marvin L., Computation: Finite and Infinite Machines Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.
4. Shoup, Richard G., "Programmable Cellular Logic Arrays", Ph.D. thesis, Carnegie-Mellon University. Pittsburg, Pa., March 1970.

APPENDIX I

Simulation of the Components: The Two State Machine

The Wire and Dead End

```
cycle 0
1               1
111111 1111111111
11111 11111111111
11111111111111111
1               1
```

```
cycle  1
1               1
1111111 111111111
111111 1111111111
11111111111111111
1               1
```

```
cycle  2
1               1
11111111 11111111
1111111 111111111
11111111111111111
1               1
```

```
cycle  3
1               1
111111111 1111111
11111111 11111111
11111111111111111
1               1
```

```
cycle  4
1               1
1111111111 111111
111111111 1111111
11111111111111111
1               1
```

```
cycle  5
1               1
11111111111 11111
1111111111 111111
11111111111111111
1               1
```

```
cycle  6
1               1
111111111111 1111
11111111111 11111
11111111111111111
1               1
```

```
cycle  7
1               1
1111111111111 111
111111111111 1111
11111111111111111
1               1
```

```
cycle  8
1               1
11111111111111 11
1111111111111 111
11111111111111111
1               1
```

```
cycle  9
1               1
111111111111111 1
11111111111111 11
11111111111111111
1               1
```

```
cycle  10
1               1
11111111111111111
111111111111111 1
11111111111111111
1               1
```

```
cycle  11
1               1
11111111111111111
11111111111111111
11111111111111111
1               1
```

Note particularly the disappearance of the signal in cycles 10 and 11.

The Fanout Element

```
cycle 0
         11111
          111
          111
          111
       ↓  111
111111 111111        1
11111 1111111111111111
1111111111   111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   1
         11111
          111
          111
          111
        ↓ 111
1111111 11111        1
111111 111111111111111
1111111111   111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   2
         11111
          111
          111
          111
        ↓ 111
11111111 1111        1
1111111 11111111111111
1111111111   111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   3
         11111
          111
          111
          111
         ↘111
111111111 111        1
11111111 1111111111111
1111111111   111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   4
         11111
          111
          111
          111
          111
1111111111111        1
111111111 111111111111
1111111111   111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   5
         11111
          111
          111
          111
          111
1111111111111        1
1111111111 11111111111
111111111  ↖ 111111111
        11   111111111
        11   111     1
       1111  111
             111
            11111
```

(Continued)

```
cycle    6
         11111
          111
          111
          111
          111
1111111111111        1
111111111 1 1111111111
1111111111 ↑ 111111111
        1    111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle    7
         11111
          111
          111
          111
          111
1111111111111        1
111111111111 111111111
1111111111  ↗111111111
        11   111111111
        1    111     1
       1111  111
             111
            11111
```

```
cycle    8
         11111
          111
          111
          111
          111 ↙
111111111111         1
11111111111 1 11111111
1111111111   111111111
        1    111111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle    9
         11111
          111
          111
          111
          11  ↙
11111111111 1  ↓     1
1111111111 111 1111111
111111111  →  11111111
        11   111111111
        1    111     1
       1111  111
             111
            11111
```

```
cycle   10
         11111
          111
          111
          11 ←
          1 1
1111111111 11  ↓     1
111111111 11111 111111
1111111111   1 1111111
        11 →  11111111
        11   111     1
       1111  111
             111
            11111
```

```
cycle   11
         11111
          111
          11  ←
          1 1
          111
11111111111 1   ↓    1
1111111111111111 11111
111111111    11 111111
        11   1 1111111
        11 →  11     1
       1111  111
             111
            11111
```

```
cycle   12
         11111
          11  ←
          1 1
          111
          111
1111111111111    ↓   1
11111111111111111 1111
1111111111   1 1 11111
        1    11 111111
        11   1 1     1
       1111 → 11
             111
            11111
```

```
cycle   13
         11111
          1 1
          111
          111
          111
1111111111111      ↓ 1
111111111111111111 111
111111111    1111 1111
        11   111111111
        1    111     1
       1111  1 1
           →  11
            11111
```

```
cycle   14
         11111
          111
          111
          111
          111
1111111111111      ↓ 1
1111111111111111111 11
1111111111   11111 111
        11   111111111
        11   111     1
       1111  111
             1 1
            11111
```

The Inside to Inside Curve

```
cycle 0
 11111
  111
  1 1
  11   ←
 1111
  111
  111
 1111    1
   1111111         1
   11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   1
 11111
  111
  111
  1 1
 111  ←
  111
  111
 1111    1
   1111111         1
   11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   2
 11111
  111
  111
  111
 11 1
  11   ←
  111
 1111    1
   1111111         1
   11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   3
 11111
  111
  111
  111
 1111
  1 1  ↙
  11
 1111    1
   1111111         1
   11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   4
 11111
  111
  111
  111
 1111
  111
  1 1 ↙
 111     1
   1111111         1
   11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   5
 11111
  111
  111
  111
 1111
  111
  111
 11 1    1
   1111111         1
↗  11111111111111111
   1      1111111111
          1111111111
          1        1
```

(Continued)

```
cycle   6
 11111
  111
  111
  111
 1111
  111
  111
 1 11    1
    111111         1
→  11111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   7
 11111
  111
  111
  111
 1111
  111
   11
 11 1    1
   1111111         1
→   1111111111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle   8
 11111
  111
  111
  111
 1111
   11
  1 1
 1111    1
   1111111         1
   1 111111111111111
   1 ↖    1111111111
          1111111111
          1        1
```

```
cycle   9
 11111
  111
  111
  111
 1111
  1 1
  111
 1111    1
   1111111         1
   11 11111111111111
   1 ↑    1111111111
          1111111111
          1        1
```

```
cycle  10
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111         1
   1 1 1111111111111
   1  ↑   1111111111
          1111111111
          1        1
```

```
cycle  11
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111         1
   1111 111111111111
   1   ↑  1111111111
          1111111111
          1        1
```

(Continued)

```
cycle  12
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111         1
   111 1 11111111111
   1   ↗  1111111111
          1111111111
          1        1
```

```
cycle  13
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111         1
   11 111 1111111111
   1    ↗ 1111111111
          1111111111
          1        1
```

```
cycle  14
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   111111 ↙        1
   1 1 1 1 111111111
   1      1111111111
          1111111111
          1        1
```

```
cycle  15
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111 ↓       1
   11111111 11111111
   1       111111111
          1111111111
          1        1
```

```
cycle  16
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111  ↓      1
   111111111 1111111
   1      1 11111111
           111111111
          1        1
```

```
cycle  17
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111   ↓     1
   1111111111 111111
   1      11 1111111
          1111111111
          1        1
```

```
cycle  18
 11111
  111
  111
  111
 1111
  111
  111
 1111    1
   1111111    ↓    1
   11111111111 11111
   1      111 111111
          1111111111
          1        1
```

The Logic Function (B <u>and</u> NOT A)

Case I, A input only

```
cycle   0
         11111
          1 1
     A →   11
          1111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   1
         11111
          111
          1 1
       →   111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   2
         11111
          111
          111
          1 11
       →   11
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   3
         11111
          111
          111
          1111
          1 1
1       ↘  11
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   4
         11111
          111
          111
          1111
          111
1         1 1
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   5
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

Note that the signal did not get through.

The Logic Function

Case II, A and B both input

```
cycle 0
         11111
          1 1
      A →  11
          1111
     B    111
1     ↓   111
111111 111111     1
11111 1111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    1
         11111
          111
          1 1
       →   111
          111
1      ↓  111
1111111 11111     1
111111 111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    2
         11111
          111
          111
          1 11
        ↘  11
1      ↘  111
11111111 1111     1
1111111 11111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    3
         11111
          111
          111
          1111
          1 1
1       ↘↘ 11
111111111 111     1
11111111 1111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    4
         11111
          111
          111
          1111
          111
1         1 1
1111111111 11     1
111111111 111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    5
         11111
          111
          111
          1111
          111
1         111
11111111111 1     1
1111111111111111111
111111111 ↖  111111
1       11   111111
        11   1    1
       1111
```

(Continued)

```
cycle   6
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       1 ←  111111
        11   1    1
       1111
```

```
cycle   7
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
1111111111111111111
111111111    111111
1       11   111111
        1    1    1
       1111
```

```
cycle   8
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle   9
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

Note again that the signal did not reach the output, although a spurious signal did enter the short leg, but was eventually anihilated.

The Logic Function (B <u>and</u> NOT A)

Case III, B input only

```
cycle 0
         11111
          111
          111
          1111
          111
1     ↓   111
111111 111111      1
11111 1111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    1
         11111
          111
          111
          1111
          111
1      ↓  111
1111111 11111      1
111111 111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    2
         11111
          111
          111
          1111
          111
1       ↓ 111
11111111 1111      1
1111111 11111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    3
         11111
          111
          111
          1111
          111
1       ↘ 111
111111111 111      1
11111111 1111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    4
         11111
          111
          111
          1111
          111
1         111
1111111111111      1
111111111 111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

```
cycle    5
         11111
          111
          111
          1111
          111
1         111
1111111111111      1
1111111111 11111111
111111111  ↑ 111111
1       11   111111
        11   1    1
       1111
```

(Continued)

```
cycle   6
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
111111111 1 1111111
1111111111 ↑ 111111
1       1    111111
        11   1    1
       1111
```

```
cycle   7
         11111
          111
          111
          1111
          111
1         111
1111111111111     1
111111111111 111111
1111111111   111111
1       11   111111
        1    1    1
       1111
```

```
cycle   8
         11111
          111
          111
          1111
          111
1         111
111111111111 ↙    1
11111111111 1 11111
1111111111   111111
1       1    111111
        11   1    1
       1111
```

```
cycle   9
         11111
          111
          111
          1111
          111
1         11
11111111111 1 ↓   1
1111111111 111 1111
111111111     11111
1       11   111111
        1    1    1
       1111
```

```
cycle  10
         11111
          111
          111
          1111
          11
1         1 1
1111111111 11  ↓  1
111111111 11111 111
1111111111   1 1111
1       11    11111
        11   1    1
       1111
```

```
cycle  11
         11111
          111
          111
          1111
          1 1
1         111
11111111111 1   ↓ 1
1111111111111111 11
111111111    11 111
1       11   111111
        11   1    1
       1111
```

(Continued)

```
cycle  12
         11111
          111
          111
          1111
          111
1         111
1111111111111   ↓ 1
11111111111111111 1
1111111111   111 11
1       1    111111
        11   1    1
       1111

cycle  13
         11111
          111
          111
          1111
          111
1         111
1111111111111   ↓ 1
1111111111111111111
111111111    1111 1
1       11   111111
        1    1    1
       1111

cycle  14
         11111
          111
          111
          1111
          111    ↓
1         111
1111111111111     1
1111111111111111111
1111111111   111111
1       11   111111
        11   1    1
       1111
```

This time a signal gets through to the output.

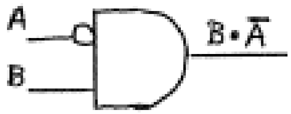
The Clock (Simulated One Clock Period)

```
cycle 0
         1      1
         11111111
 1        1111111
 111111111 111111
1111111 1 ↑     1
      ↗ 1

cycle    1
         1      1
          1111111
 1       1 111111
 1111111111 11111
111111 11   ↑   1
      ↑ 1

cycle    2
         1      1
         11111111
 1       11 11111
 11111111111 1111
11111 1 1   ↑   1
     ↑  1

cycle    3
         1      1
         11111111
 1       111 1111
 111111111111 111
1111 1111    ↑  1
    ↑   1

cycle    4
         1      1
         11111111
 1       1111 111
 1111111111111 11
111 1 111     ↑ 1
   ↑    1

cycle    5
         1      1
         11111111
 1       11111 11
 11111111111111 1
11 111 11      ↑1
  ↑     1

cycle    6
         1      1
         11111111
 1       111111 1
 1111111111111111
1 1 1 1 1       1
 ↑      1

cycle    7
         1      1
         11111111
 1       11111111
→ 111111111111111
111111111       1
        1

cycle    8
         1      1
         11111111
 1↙      11111111
 1 11111111111111
111111111       1
        1

cycle    9
         1      1
         11111111
 1 ↓     11111111
 11 1111111111111
111111111       1
        1

cycle   10
         1      1
         11111111
 1  ↓    11111111
 1 1 111111111111
111111111       1
        1

cycle   11
         1      1
         11111111
 1   ↓   11111111
 1111 11111111111
111111111       1
        1

cycle   12
         1      1
         11111111
 1    ↓  11111111
 111 1 1111111111
111111111       1
        1

cycle   13
         1      1
         11111111
 1     ↓ 11111111
 11 111 111111111
111111111       1
        1

cycle   14
         1      1
         11111111
 1     ↘ 11111111
 1 1 1 1 11111111
111111111       1
        1

cycle   15
         1      1
         11111111
 1       11111111
 11111111 1111111
11111111  ↖     1
        1

cycle   16
         1      1
         11111111
 1        1111111
 111111111 111111
1111111 1 ↑     1
      ↗ 1

cycle   17
         1      1
          1111111
 1       1 111111
 1111111111 11111
111111 11  ↑    1
      ↑ 1
```

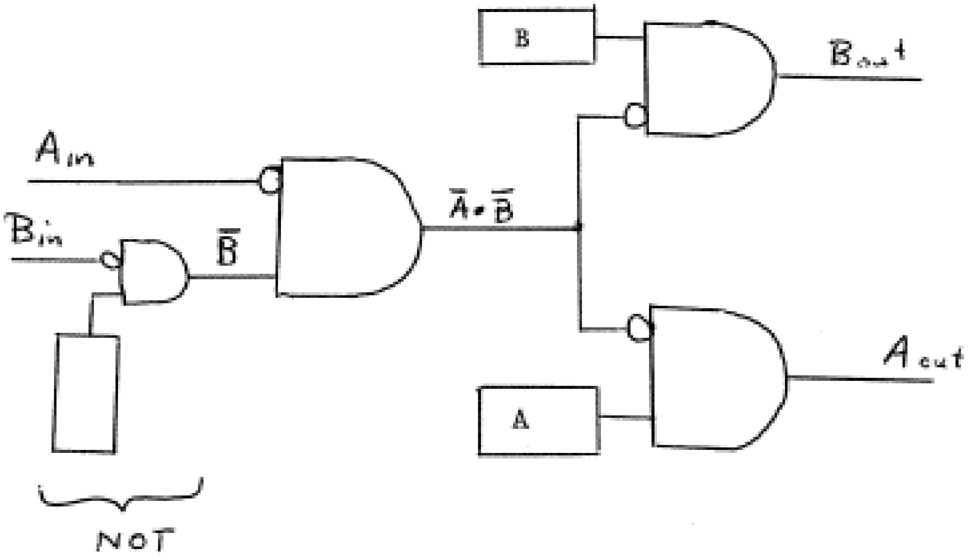
APPENDIX II

Another Crossover for the Two State Cellular Automaton

The logic function B _and_ NOT A is represented by the following symbol.



The clocks are represented by boxes. The crossover is constructed from these elements.



The clocks labeled A and B are out of phase with each other and have twice the period of the other clock. This entire configuration has been simulated for the cases A input, B input and neither input. This component is active even when there are no inputs. The clocks are running and there is usually a signal in the internal wire. This signal is destroyed by an input allowing one of the clocks labeled A and B to get through to the output.

APPENDIX III

Simulation of the Components: The Three State Cellular Automaton

The Wire and Dead-End

```
cycle 0

22222222222222222
2222222 122222222
22222222222222222
      →

cycle 1

22222222222222222
22222222 12222222
22222222222222222

cycle 2

22222222222222222
222222222 1222222
22222222222222222

cycle 3

22222222222222222
2222222222 122222
22222222222222222

cycle 4

22222222222222222
22222222222 12222
22222222222222222
```

```
cycle 5

22222222222222222
222222222222 1222
22222222222222222

cycle 6

22222222222222222
2222222222222 122
22222222222222222

cycle 7

22222222222222222
22222222222222 12
22222222222222222

cycle 8              ↓

22222222222222222
222222222222222 2
22222222222222222

cycle 9

22222222222222222
22222222222222222
22222222222222222
```

The Fanout

```
cycle 0
    222
    222
    222
→   222
22222222222
2 122222222
22222222222
    222
    222
    222
    222
```

```
cycle    1
    222
    222
    222
    222
22222222222
22 12222222
22222222222
    222
    222
    222
    222
```

```
cycle    2
    222
    222
    222
    222
22222222222
222 1222222
22222222222
    222
    222
    222
    222
```

```
cycle    3
    222
    222
    222
    222
22221222222
2222 122222
22221222222
    222
    222
    222
    222
```

```
cycle    4
    222
    222
    222
    222
2222 122222
22222 12222
2222 122222
    222
    222
    222
    222
```

```
cycle    5
    222
    222
    222
    212
22222 12222
222222 1222
22222 12222
    212
    222
    222
    222
```

```
cycle    6
    222
    222
    212
    2 2
22222  2222
2222222 122
22222  2222
    2 2
    212
    222
    222
```

```
cycle    7
    222
  ↑ 212
    2 2
    222 →
22222 12222
22222222 12
22222 12222
    222
  ↓ 2 2
    212
    222
```

```
cycle    8
    222
    2 2
    222
    222
222222 2222
222222222 2
222222 2222
    222
    222
    2 2
    222
```

```
cycle    9
    222
    222
    222
    222
22222222222
22222222222
22222222222
    222
    222
    222
    222
```

Two Inputs Give Two Outputs.

```
cycle 0
    222
  ↓ 2 2
    212
→   222
22222222222
2 122222222
22222222222
    222
    222
    222
    222
```

```
cycle    1
    222
    222
    2 2
    212
22222222222
22 12222222
22222222222
    222
    222
    222
    222
```

```
cycle    2
    222
    222
    222
    2 2
22222122222
222 1222222
22222222222
    222
    222
    222
    222
```

```
cycle    3
    222
    222
    222
    222
22221 12222
2222 122222
22221222222
    222
    222
    222
    222
```

```
cycle    4
    222
    222
    222
    222
2222 2 2222
22222 12222
2222 122222
    222
    222
    222
    222
```

```
cycle    5
    222
    222
    222
    222
22222222222
22222  1222
22222 12222
    212
    222
    222
    222
```

```
cycle    6
    222
    222
    222
    222
22222222222
22222 2 122
222222 2222
    2 2
    212
    222
    222
```

```
cycle    7
    222
    222
    222
    222  →
22222222222
22222222 12
22222222222
    222
  ↓ 2 2
    212
    222
```

```
cycle    8
    222
    222
    222
    222
22222222222
222222222 2
22222222222
    222
    222
    2 2
    222
```

```
cycle    9
    222
    222
    222
    222
22222222222
22222222222
22222222222
    222
    222
    222
    222
```

Three Inputs Are Mutually Anihilated.

```
cycle 0
    222
  ↓ 2 2
    212
    222
22222222222
2 122222222
22222222222
 →  222 ↑
    212
    2 2
    222
```

```
cycle    1
    222
    222
    2 2
    212
22222222222
22 12222222
22222222222
    212
    2 2
    222
    222
```

```
cycle    2
    222
    222
    222
    2 2
22222122222
222 1222222
22222122222
    2 2
    222
    222
    222
```

```
cycle    3
    222
    222
    222
    222
22221 12222
2222  22222
22221 12222
    222
    222
    222
    222
```

```
cycle    4
    222
    222
    222
    222
2222 2 2222
22222 22222
2222 2 2222
    222
    222
    222
    222
```

```
cycle    5
    222
    222
    222
    222
22222222222
22222222222
22222222222
    222
    222
    222
    222
```

The Special Wire or Tape

```
cycle 0

222222222
22222 1222
222222222
```

```
cycle   1

222222222
222222 122
222222222
```

```
cycle   2

222222222
2222222 12
222222222
```

```
cycle   3

222222222
22222222 2
222222222
```

```
cycle   4

222222222
2222222221
222222222
```

```
cycle   5

2222222222
22222222112
2222222222
```

```
cycle   6

2222222222
22222221  2
2222222222
```

```
cycle   7

2222222222
2222221 211
2222222222
```

```
cycle   8

22222222222
222221 22 22
22222222222
```

```
cycle   9

22222222222
22221 222222
22222222222
```

←echo

```
cycle  10

22222222222
2221 2222222
22222222222
```

Note that the wire gets longer and that an echo signal is generated.

Meeting Signals Cancel.

```
cycle 0

2222222222222222
2222 12221 22222
2222222222222222


cycle    1

2222222222222222
22222 121 222222
2222222222222222


cycle    2

2222222222222222
222222 2 2222222
2222222222222222


cycle    3

2222222222222222
2222222222222222
2222222222222222
```

The Transition Rules For the Three State Cellular Automaton

These Rules are Listed in the form "(CNESWR)" where C represents the current state, N the state of the North neighbor, E the East neighbor, S the South and W the West neighbor. The result state is represented by R. Only transition rules are listed, i.e., C is different from R. Also the North, East, South, and West neighbors can be rotated or flipped, e.g., N can be swapped with S.

```
( 12222)
( 22222)
( 1   2)
( 12  2)
( 21  2)
( 12 22)
( 222 1)
( 11122)
( 21 12)
( 112 2)
( 211 2)
(1 222 )
(11222 )
(11   2)
(11   2)
(122   )
(212221)
(2    1)
(211221)
(21112 )
```

APPENDIX IV

Simulation of the Components: The Four State Cellular Automaton

The Wire and Dead-End

```
cycle 0

 xxxxxxxxxx
 x   12
 xxxxxxxxxx



cycle    1

 xxxxxxxxxx
 x  12
 xxxxxxxxxx



cycle    2

 xxxxxxxxxx
 x 12
 xxxxxxxxxx



cycle    3

 xxxxxxxxxx
 x12
 xxxxxxxxxx



cycle    4

 xxxxxxxxxx
 x2
 xxxxxxxxxx



cycle    5

 xxxxxxxxxx
 x
 xxxxxxxxxx



cycle    6

 xxxxxxxxxx
 x
```

The Fanout Junction

```
cycle 0
    xxx
    x x
    x x
    x x  ↓
xxxxx xxxxx
x    x  12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 1
    xxx
    x x
    x x
    x x
xxxxx xxxxx
x    x 12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 2
    xxx
    x x
    x x
    x x
xxxxx xxxxx
x    x12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 3
    xxx
    x x
    x x
    x x
xxxxx xxxxx
x    22
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 4
    xxx
    x x
    x x
    x x
xxxxx xxxxx
x    12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 5
    xxx
    x x
    x x
    x x
xxxxx1xxxxx
x   12
xxxxx1xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle 6
    xxx
    x x
    x x
    x1x
xxxxx2xxxxx
x  12x
xxxxx2xxxxx
    x1x
    x x
    x x
    xxx
```

```
cycle 7
    xxx
    x x
    x1x
    x2x
xxxxx xxxxx
x 12 x
xxxxx xxxxx
    x2x
    x1x
    x x
    xxx
```

```
   cycle 8
       xxx
     → x1x
       x2x
       x x
   xxxxx xxxxx
→  x12  x
   xxxxx xxxxx
       x x
       x2x
     → x1x
       xxx
```

```
cycle 9
    xxx
    x2x
    x x
    x x
xxxxx xxxxx
x2   x
xxxxx xxxxx
    x x
    x x
    x2x
    xxx
```

```
cycle 10
    xxx
    x x
    x x
    x x
xxxxx xxxxx
x    x
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

Two Input, Two Output

```
cycle 0
    x x
    x2x
    x1x
    x x
xxxxx xxxxx
x    x  12
xxxxx xxxxx
    x x
    x x
    x x
    xxx

cycle    1
    x x
    x x
    x2x
    x1x
xxxxx xxxxx
x    x 12
xxxxx xxxxx
    x x
    x x
    x x
    xxx

cycle    2
    x x
    x x
    x x
    x2x
xxxxx1xxxxx
x    x12
xxxxx xxxxx
    x x
    x x
    x x
    xxx

cycle    3
    x x
    x x
    x x
    x x
xxxxx2xxxxx
x    12
xxxxx xxxxx
    x x
    x x
    x x
    xxx

cycle    4
    x x
    x x
    x x
    x x
xxxxx xxxxx
x   12
xxxxx1xxxxx
    x x
    x x
    x x
    xxx

cycle    5
    x x
    x x
    x x
    x x
xxxxx xxxxx
x  12x
xxxxx2xxxxx
    x1x
    x x
    x x
    xxx

cycle    6
    x x
    x x
    x x
    x x
xxxxx xxxxx
x 12 x
xxxxx xxxxx
    x2x
    x1x
    x x
    xxx

cycle    7
    x x
    x x
    x x
    x x
xxxxx xxxxx
x12  x
xxxxx xxxxx
    x x
    x2x
    x1x
    xxx

cycle    8
    x x
    x x
    x x
    x x
xxxxx xxxxx
x2   x
xxxxx xxxxx
    x x
    x x
    x2x
    xxx

cycle    9
    x x
    x x
    x x
    x x
xxxxx xxxxx
x    x
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

Three Simultaneous Signals Cancel.

```
cycle 0
    x x
    x2x
    x1x
    x x
xxxxx xxxxx
 21  x  12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle    1
    x x
    x x
    x2x
    x1x
xxxxx xxxxx
  21 x 12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle    2
    x x
    x x
    x x
    x2x
xxxxx1xxxxx
   21x12
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle    3
    x x
    x x
    x x
    x x
xxxxx2xxxxx
    2x2
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

```
cycle    4
    x x
    x x
    x x
    x x
xxxxx xxxxx
     x
xxxxx xxxxx
    x x
    x x
    x x
    xxx
```

The Single Wing Signal

```
cycle 0

     1
xxxxxx1xxxxxxx


cycle    1

      1
xxxxxxx1xxxxxx


cycle    2

       1
xxxxxxxx1xxxxx


cycle    3

        1
xxxxxxxxx1xxxx


cycle    4

         1
xxxxxxxxxx1xxx


cycle    5

          1
xxxxxxxxxxx1xx
```

Colliding Single Wings Form an Arm Bud

```
cycle 0
 x
 x1
 1
 x
 1
 x1
 x
```

```
cycle   1
 x
 x
 x1
 x
 x1
 x
 x
```

```
cycle   2
 x
 x
 x
 x2
 x
 x
 x
```

```
cycle   3
 x
 x
 xx
 x  x
 xx
 x
 x
```

```
cycle   4
 x
 x
 xx
 x1x
 xx
 x
 x
```

```
cycle   5
 x
 x
 x1
 xx2
 x1
 x
 x
```

```
cycle   6
 x
 x
 x
 x1  x
 x
 x
 x
```

```
cycle   7
 x
 x
 x
 x  xx
 x
 x
 x
```

```
cycle   8
 x
 x
 x
 x  xx
 x
 x
 x
```

Filling the Gap

```
cycle 0
 x
 x1
 1
 x xx
 1
 x1
 x

cycle   1
 x
 x
 x1
 x xx
 x1
 x
 x

cycle   2
 x
 x
 x
 xxxx
 x
 x
 x

cycle   3
 x
 x
 x
 xxxx
 x
 x
 x
```

Lengthening the Arm

```
cycle 0
 x
 x1
 1
 xxxx
 1
 x1
 x

cycle   1
 x
 x
 x1
 xxxx
 x1
 x
 x

cycle   2
 x
 x
 x
 x2xx
 x
 x
 x

cycle   3
 x
 x
 xx
 x1xx
 xx
 x
 x

cycle   4
 x
 x
 x1
 xx1x
 x1
 x
 x
```

```
cycle   5
 x
 x
 x 1
 xxx2
 x 1
 x
 x

cycle   6
 x
 x
 x
 xx1 x
 x
 x
 x

cycle   7
 x
 x
 x
 x1 xx
 x
 x
 x

cycle   8
 x
 x
 x
 x xxx
 x
 x
 x

cycle   9
 x
 x
 x
 x xxx
 x
 x
 x
```

Lengthening of the Arm
(Notice the Echo Signal)

```
cycle 0
 x
 x1
 1
 xxxxxxxxx
 1
 x1
 x
```

```
cycle   1
 x
 x
 x1
 xxxxxxxxx
 x1
 x
 x
```

```
cycle   2
 x
 x
 x
 x2xxxxxxx
 x
 x
 x
```

```
cycle   3
 x
 x
 xx
 x1xxxxxxx
 xx
 x
 x
```

```
cycle   4
 x
 x
 x1
 xx1xxxxxx
 x1
 x
 x
```

```
cycle   5
 x
 x
 x 1
 xxx1xxxxx
 x 1
 x
 x
```

```
cycle   6
 x
 x
 x  1
 xxxx1xxxx
 x  1
 x
 x
```

```
cycle   7
 x
 x
 x   1
 xxxxx1xxx
 x   1
 x
 x
```

```
cycle   8
 x
 x
 x    1
 xxxxxx1xx
 x    1
 x
 x
```

```
cycle   9
 x
 x
 x     1
 xxxxxxx1x
 x     1
 x
 x
```

```
cycle  10
 x
 x
 x      1
 xxxxxxxx2
 x      1
 x
 x
```

```
cycle  11
 x
 x
 x
 xxxxxxx1 x
 x
 x
 x
```

```
cycle  12
 x
 x
 x
 xxxxxx1 xx
 x     ↑
 x
 x
```

```
cycle  13
 x
 x
 x
 xxxxx1 xxx
 x    ↑
 x
 x
```

```
cycle  14
 x
 x
 x
 xxxx1 xxxx
 x   ↑
 x
 x
```

```
cycle  15
 x
 x
 x
 xxx1 xxxxx
 x  ↑
 x
 x
```

```
cycle  16
 x
 x
 x
 xx1 xxxxxx
 x
 x
 x
```

Creation of the Erasing Wing by Coincidence of Single Wings and Echo Signal

```
cycle 0
 x
 x1
 1
 xxxxx1 xxx
 1
 x1
 x
```

```
cycle   1
 x
 x
 x1
 xxxx1 xxxx
 x1
 x
 x
```

```
cycle   2
 x
 x
 x
 x2x1 xxxxx
 x
 x
 x
```

```
cycle   3
 x
 x
 xx
 x11 xxxxxx
 xx
 x
 x
```

```
cycle   4
 x
 x
 x1
 x 1xxxxxxx
 x1
 x
 x
```

```
cycle   5
 x
 x
 x 1
 x2 1xxxxxx
 x 1
 x
 x
```

```
cycle   6
 x
 x
 x  1
 x   1xxxxx
 x  1
 x
 x
```

```
cycle   7
 x
 x
 x   1
 x    1xxxx
 x   1
 x
 x
```

```
cycle   8
 x
 x
 x    1
 x     1xxx
 x    1
 x
 x
```

```
cycle   9
 x
 x
 x     1
 x      1xx
 x     1
 x
 x
```

```
cycle  10
 x
 x
 x      1
 x       1x
 x      1
 x
 x
```

```
cycle  11
 x
 x
 x       1
 x        2
 x       1
 x
 x
```

```
cycle  12
 x
 x
 x         ↘
 x          x
 x
 x
 x
```

```
cycle  13
 x
 x
 x         ↘
 x          x
 x
 x
 x
```

Creation of the Single Wings from the Logic Type Signals

```
cycle 0
   x
xxxxx
 21  xxxxxxxxxxx
xxxxx
    x

cycle    1
   x
xxxxx
  21 xxxxxxxxxxx
xxxxx
    x

cycle    2
   x
xxxxx
   21xxxxxxxxxxx
xxxxx
    x

cycle    3
   x
xxxx1
    21xxxxxxxxxx
xxxxx
    x

cycle    4
   x
xxxxx1
    2x1xxxxxxxxx
xxxxx
    x

cycle    5
   x
xxxxx 1
     xx1xxxxxxxx
xxxxx
    x

cycle    6
   x
xxxxx  1
     xxx1xxxxxxx
xxxxx
    x

cycle    7
   x
xxxxx   1
     xxxx1xxxxxx
xxxxx
    x

cycle    8
   x
xxxxx    1
     xxxxx1xxxxx
xxxxx
    x

cycle    9
   x
xxxxx     1
     xxxxxx1xxxx
xxxxx
    x
```

Three Cases of the Depositing of an "x" State

```
cycle 0                  cycle 0               cycle 0

  1                        1     x               1     x
   1xxxx x                  1xxxx                 1xxxx x
  1                        1                     1


cycle   1                cycle   1             cycle   1

   1                        1    x                1    x
    1xxx x                   1xxx                  1xxx x
   1                        1                     1


cycle   2                cycle   2             cycle   2

    1                        1   x                 1   x
     1xx x                    1xx                   1xx x
    1                        1                     1


cycle   3                cycle   3             cycle   3

     1                        1  x                  1  x
      1x x                     1x                    1x x
     1                        1                     1


cycle   4                cycle   4             cycle   4

      1                        1 x                   1 x
       2 x                      2                     2 x
      1                        1                     1


cycle   5                cycle   5             cycle   5

                                 x                     x
        xx                       x                     xx
       ↗                        ↗                     ↗


cycle   6                cycle   6             cycle   6

                                 x                     x
        xx                       x                     xx
```

The Capture of an Erasing Wing Signal to Form a New Logic Signal

(Note that the Capture Mechanism Becomes a Dead End.)

```
cycle  0
  xxx
  x x
  x x
  x x
 xx xx
xx x xx
xx x xx
   1
  1 1

cycle    1
  xxx
  x x
  x x
  x x
 xx xx
xx x xx
xx 1 xx
  1 1

cycle    2
  xxx
  x x
  x x
  x x
 xx xx
xx 2 xx
xx1 1xx

cycle    3
  xxx
  x x
  x x
  x x
 xx xx
xx2 2xx
xx   xx

cycle    4
  xxx
  x x
  x x
  x x
 xx xx
xx212xx
xxx xxx

cycle    5
  xxx
  x x
  x x
  x x
 xx1xx
xx 2 xx
xxx1xxx

cycle    6
  xxx
  x x
  x x
  x1x
 xx2xx
xx   xx
xx121xx

cycle    7
  xxx
  x x
  x1x
  x2x
 xx xx
xx 1 xx
xxx xxx
   x

cycle    8
  xxx
  x1x
  x2x
  x x
 xx xx
xx1 1xx
xxx1xxx
   x

cycle    9
  xxx
  x2x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx
   2

cycle   10
  xxx
  x x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx
  x x
   x

cycle   11
  xxx
  x x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx
  x1x
   x

cycle   12
  xxx
  x x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx
  1x1
   2

cycle   13
  xxx
  x x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx
   1

   x

cycle   14
  xxx
  x x
  x x
  x x
 xx xx
xxx xxx
xxxxxxx

   x
   x
```

The Transition Rules for the Four State Cellular Automaton

These rules are listed in the form "(CNESWR)". See Appendix III for further explanation.

The states are represented by the symbols "x", "1", "2", and the blank space.

```
( x1x 1)          ( 1x1 2)
(1x2x 2)          (2x    )
(2x1x  )          ( xxxx1)
( 1xxx1)          (xx1  1)
(x1   2)          (x1x  1)
(22   1)          (11x   )
(12   2)          (1x1   )
(12xxx2)          (121x1x)
(2111 x)          (21   x)
(21xxx )          (x1x1x2)
(2xxx  )          (2x x 1)
(x11  1)          (1xxxxx)
(122  2)          (1x x x)
(211  x)          ( 11x 1)
(2xxx  )          ( x11 1)
(x1 x 1)          (x21x11)
(1     )          (1xx   )
(1x    )          ( 1x1xx)
( 11  1)          ( 111x2)
(x1   2)          (x2 1 1)
( 2   x)          (11xxx )
(2     )          (1xxx x)
( 121x1)          (x1 1 1)
( 1 x x)          (12x  x)
( 111  )          (1x2  x)
( 121  )          (12 x x)
( xx2 x)          ( 21xx2)
( 2xx x)          ( 12xx2)
( 2 x x)          ( 2 2 1)
( 2x  x)          (12 2 2)
( x2  x)          (21 1  )
```

DISCUSSION

III. Practical Application

Perhaps the predominant application of cellular automata theory will be in circuits with a self wiring capability. A large scale integrated circuit forming a large array of cells could contain an initial configuration with construction power. When activated, it would proceed to wire up any desired circuit, (as indicated by information built into the configuration or sent into it after activation) detecting and ignoring bad cells. Such a circuit could actually be a complete parallel or serial digital computer on one slice. Various regions of the slice could have different transition rules if, for example, one type cell proved more efficient for memory and another for logic functions. Variations include having no initial configuration wired in, but sending inputs in to build a constructing mechanism.

The main point is that by allowing bad areas to appear on the slice without destroying the total slice, much cheaper and more complex circuitry could be used. Shoup (4) has considered the design of integrated circuits used as cellular arrays. Not concerning himself with universality in particular, he was more interested in efficient implementations of practical operations.