INFORMATION PROCESSING AND TRANSMISSION IN CELLULAR AUTOMATA

Edwin R. Banks

January 1971

PROJECT MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge Massachusetts 02139

INFORMATION PROCESSING AND TRANSMISSION IN CELLULAR AUTOMATA

Edwin Roger Banks

Abstract

Submitted to the Department of Mechanical Engineering
on January 15, 1971
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

A cellular automaton is an iterative array of very simple identical information processing machines called cells. Each cell can communicate with neighboring cells. At discrete moments of time the cells can change from one state to another as a function of the states of the cell and its neighbors. Thus on a global basis, the collection of cells is characterized by some type of behavior.

The goal of this investigation was to determine just how simple the individual cells could be while the global behavior achieved some specified criterion of complexity--usually the ability to perform a computation or to reproduce some pattern.

The chief result described in this thesis is that an array of identical square cells (in two dimensions), each cell of which communicates directly with only its four nearest edge neighbors and each of which can exist in only two states, can perform any computation. This computation proceeds in a straight forward way.

A configuration is a specification of the states of all the cells in some area of the iterative array. Another result described in this thesis is the existence of a self-reproducing configuration in an array of four-state cells, a reduction of four states from the previously known eight-state case.

The technique of information processing in cellular arrays involves the synthesis of some basic components. Then the desired behaviors are obtained by the interconnection of these components. A chapter on components describes some sets of basic components.

Possible applications of the results of this investigation, descriptions of some interesting phenomena (for vanishingly small cells), and suggestions for further study are given later.

Thesis Supervisor: Edward Fredkin

Title: Professor of Electrical Engineering

ACKNOWLEDGMENTS

I wish to thank Professor Edward Fredkin, the supervisor of this thesis, for introducing me to the topic and for providing expert supervision especially pertaining to the direction which this investigation has followed. I wish also to thank Professor Thomas B. Sheridan, the thesis committee chairman, and Professors Marvin Minsky and Henry Paynter, the other members of my thesis committee, for their criticism and guidance of this work. To many other Project MAC personnel my gratitude goes for stimulating conversations and for providing the computation facilities.

This work was supported by the Artificial Intelligence Laboratory of Project MAC, 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.

TABLE OF CONTENTS

Abstract 2

Acknowledgments. 3

I. Introduction 5

Justification and Goals of this Research. 5

Results 6

Definition of Some Terms. 7

II. Previous Work. 9

III. Components and Techniques. 12

Information Transmission. 12

The General Technique 13

IV. An Early Three-State Universal Computer. 15

V. The Infinitely Configured Two-State Cellular Space . . . 21

VI. The Three-State Finite Universal Computer. 27

VII. The Finite Nine-Neighbor Universal Computer. 31

VIII. The Self-Reproducing Universal Computer Constructor. . . 35

IX. Universal One-Dimensional Cellular Space 43

X. Discussion 44

Starting and Halting. 44

The Continuous Space. 44

Other Applications of Cellular Automata Theory. . . 45

XI. Suggestions for Further Study. 47

XII. Summary. 49

Glossary 50

Appendices 53

Appendix I. 54

Appendix II 72

Appendix III. 80

Appendix IV 87

Biographical Note. 99

References 100

I. INTRODUCTION

Justification and Goals of this Research.

In the future I expect to see the development of a new, highly interdisciplinary field of study in this branch of automata theory known as cellular automata. Applications are apparent, of course, to parallel processing computers. Codd (1968)* mentions the benefits of mass production due to homogeneity of components, the ability to incrementally add computation power, and self-organizing machines. Zuse (1969) and others have discussed cellular models of physical phenomena and of physics itself. Von Neumann was planning a neuron-like "excitation-threshold-fatigue" model of his cellular automaton in connection with the nervous system. These are just a few areas of application.

The usefulness of a cellular model of something often will depend on the simplicity and uniformity of the cells. Similarly, our models of nature appear more useful when they consist of large numbers of simple basic components instead of the opposite. For example, physicists were once happy to believe the universe to be composed of an enormous number of just three basic components: protons, electrons and neutrons. Their search for the quark is an attempt to return to this condition. Biological cells are another example.

The goal of this investigation was to determine how simple the cells could be while the global behavior of the cellular array achieved specified degrees of complexity as measured by various capacities for universality. (This term is discussed later.) In particular, we have sought universal cells with as few states and neighbors as possible.

* References are listed by author and date.

Results.

The chief result was the discovery of a two-dimensional array of two-state cells, each of which could communicate with its four edge-neighbors (forming a five-cell neighborhood), which possessed the ability to perform any computation (thus forming a universal computer *). This result was obtained by showing that the different parts of the array could correspond to the parts of a digital computer with infinite memory capacity and also to a Turing machine. However, the method involved an initial configuration of states with an infinite number of cells in each state.

The undesirable requirement of an infinite initial configuration was eliminated in another cellular automaton by allowing another state per cell. This three-state universal computer** needed only a finite initial configuration. Increases in information storage space were obtained by the configuration's "growing" into the surrounding quiescent space.

Another successful approach to the finite universal computer was by the enlargement of the neighborhood by allowing the four corner-neighbors to also communicate with each cell. The resulting two-state, nine-neighbor automaton functioned similarly to the above by growing into the surrounding space.

A classical endeavor in this area is to find a configuration in an array of simple cells which can reproduce itself by growing a construction-arm into the area surrounding the configuration. Such a configuration is usually also required to be a universal computer. A four-state, five-neighbor self-reproducing universal computer was found and described--a reduction from the previously known eight-state case.

* In this chapter underlined terms are defined in the Glossary.

** The use of such terms as "three-state universal computer" should be interpreted as "universal computer in an array of three-state cells". Obviously the total array can have an infinite number of states.

Definition of Some Terms.

At this point the reader can consider cellular automata to be large, two-dimensional iterative arrays of simple, finite-state machines, each of which communicates with its four nearest neighbors. These finite-state machines are called cells. In the following illustration, small numbers represent the state of each cell.

0	2	0	3	0	0	0	0	0
1	0	0	1	1	1	0	2	0
0	0	0	1	1	0	0	0	0
0	0	1	3	0	0	0	0	0

A Portion of a Cellular Array at Time T

At discrete moments of time ...,t, t+1, t+2, ... the cells are allowed to change state according to a set of state transition rules. A typical such rule is illustrated and explained below.

$$\begin{array}{ccc} & 1 & \\ 1 & 0 & 2 \longrightarrow 3 \\ & 1 & \end{array}$$

A Typical Transition Rule

The above rule states that every cell in state "0" when surrounded by cells in states "1", "1", "1" and "2" will assume state "3" during the next time period. Notice that the state of a cell at time T+1 depends only on the states of the cell and of its four neighbors at time T. A set of such transition rules defines the cellular space. Also, if a cell ever exists in a local state configuration not covered by a transition rule, then the cell does not change state during the next time step. This convention allows us to not list the non-transition rules.

Since the cellular spaces discussed in this thesis are isotropic, the following four transition rules imply each other. Notice that

$$\begin{array}{ccc} & 1 & \\ 1 & 0 & 2 \longrightarrow 3 \\ & 1 & \end{array} \qquad \begin{array}{ccc} & 2 & \\ 1 & 0 & 1 \longrightarrow 3 \\ & 1 & \end{array} \qquad \begin{array}{ccc} & 1 & \\ 2 & 0 & 1 \longrightarrow 3 \\ & 1 & \end{array} \qquad \begin{array}{ccc} & 1 & \\ 1 & 0 & 1 \longrightarrow 3 \\ & 2 & \end{array}$$

Equivalent Transition Rules

the left hand side of each rule is merely a rotated image of the others implying a rotation-symmetry, one degree of isotropy.

II. PREVIOUS WORK

John von Neumann and E. F. Codd have been the chief investigators of this area. Their books deal with the notions of computation universality and construction universality. The former notion applies to cellular configurations which can perform any computable computation. The results of a computation are interpreted from the cellular configuration. Construction is a term for configurations that can "construct" other configurations. Since Moore's Garden-of-Eden theorem (Moore 1961) implies the non-constructability of some configurations, we will not require the ability to construct any configuration. Rather, the term universal will apply if there exists some configuration which can construct a new configuration, this new configuration being able to compute any specified computation. Such a universal constructor must also be able to construct a copy of itself.

Von Neumann's primary interest was in finding a computation universal cellular space with self-reproducing configurations (von Neumann 1966). His particular solution functioned by growing an arm which could construct a new passive configuration. An activation signal sent along the arm started the new configuration operating. His set of transition rules involved a five-neighbor neighborhood with twenty-nine states per cell. The set of transition rules was not isotropic.

E. F. Codd (1968) reduced the required number of states to only eight for the same task. Although his rules possessed no preferred direction, they did possess a preferred rotation. To explain, his rules allowed the automaton to send a symmetric signal down a straight arm with the result that when the signal reached the end of an arm, a right-hand corner would appear--i.e. the arm bent to the right.

Both von Neumann and Codd worked with the following requirements:

1 The cellular machines are initially configured in a finite region of the cellular space. In other words, only a finite number of non-quiescent (see Glossary) cells existed in the initial configuration.

2 Construction of new configurations by a configuration is performed into an initially quiescent region of space. This requirement rules out construction by the assembling of parts.

Codd (1968) has also shown that two states are sufficient for universality if the neighborhood is allowed to be greatly increased. In particular, he has shown that an array of two-state cells with eighty-five neighbors each can simulate his universal eight-state cells.

Although it is often true that the process of simplification (here the search for cells with small state count) yields some other aspect becoming more complex as in the case of trying to find minimal Turing machines, here the opposite is true. The automata described here are built around a very small number of configurations (including the universal logic function). Also the two-state automaton described later, for example, has only three transition rules in comparison to the hundreds of rules for the Codd automaton using eight-state cells.

Smith (1968, 1969) has discovered some interesting theorems concerning the simulations of Turing machines by cellular automata. Although a thirteen-state, five-neighbor space described by him is five more states than used by Codd, it can simulate a Turing machine in only two rows of the array. In one dimension, with p representing the neighborhood size and q the number of states per cell, Smith has found the following one-dimensional computation universal p x q spaces: 2x40, 3x18, 6x7, 8x5, 9x4, 12x3, and 14x2. Notice the existence of a universal space with only two-neighbor cells.

Shoup (1970) has considered the design of integrated circuits for use in cellular arrays. Not concerning himself with universality, he was more interested in efficient implementations of practical operations.

Hennie (1961) and Moore (1961) have studied the limitations and capabilities of cellular arrays without regard to the particular function. Amoroso, Leiblien and Yamada (1969) have attempted the definition of a mathematical framework for the formulation of questions concerning uniform arrays of finite-state machines.

Conway (1970) has invented a game called Life which employes a nine-neighbor, two-state cellular space. His game is described in the mathematical games section of a recent issue of the Scientific American. William Gosper of MIT has shown the computation universality of this space. Beyer (1969) has studied the use of cellular arrays for pattern recognition. Minsky's and Papert's perceptrons (1969) are related to this area. Zuse (1969) has considered the modeling of the laws of physics with cellular automata. Smith (1970) has related formal languages and cellular automata.

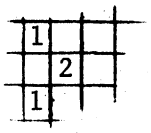
Considering self-reproducing patterns apart from computation universality, Edward Fredkin of MIT has described the following interesting cellular space. If the states are 0 and 1 and if the result state is obtained by taking the sum of the neighbors, modulo 2, then every initial configuration will reproduce copies of itself about the initial configuration. Terry Winograd (1970) generalized this result showing that any neighborhood, not just the four nearest neighbors, and any number of dimensions still yields the same results. Further, he has shown that if there are p states 0, 1, 2, ..., 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.

Further discussion of some of these areas will follow later.

III. COMPONENTS AND TECHNIQUES

Information Transmission.

The process of computation requires the capacity to move or transmit information. One type of transmission is the self-propagating signal. For example, in the following figure, a signal made up of three cells in states 1 and 2 will propagate to the right given a suitable set of transition rules.



Signal Propagation

We will use the convention of letting both the blank and zero represent a cell in the quiescent state. For the above signal to move to the right, the cells ahead of the signal must become non-quiescent and the cells currently covered by the signal must become quiescent. Thus the following transition rules are needed for the propagation to occur. (A five-neighborhood is assumed.)

```
  0                0                0                0
0 2 0 ⟶ 0        0 1 0 ⟶ 0        1 0 0 ⟶ 1        2 0 0 ⟶ 2
  0                0                2                0
```

Rules for Destruction of Signal Rules for Creation of New Signal*

A disadvantage of this type of information transmission is that any two-state, five-neighbor version of this "wireless" signal will necessarily propagate to the sides also, due to the necessity of the following transition rule. The reason for this is that if a zero in

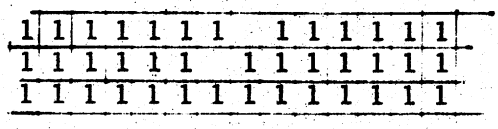
```
  0
0 0 1 ⟶ 1
  0
```

A Necessary Transition Rule

front of the signal is to become a one, then this rule is needed. However, this same rule also causes zeros adjacent to the sides and back of the signal to become ones.

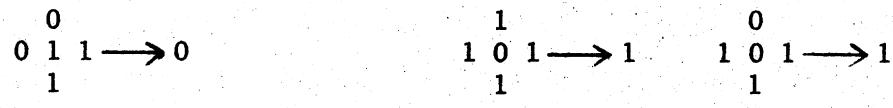
* The left-hand side of any rule can be rotated 90, 180 or 270 degrees or mirror-reflected and rotated giving another valid rule due to the assumed isotropy of all automata described herein.

This disadvantage does not exist for wired propagations. The wire is thus one of the components of each of our cellular automata. The following wire configuration is the one actually used for an infinitely configured universal computer described later.



A Two-state Wire and Signal

The gap in the wire is the signal. For this signal to propagate to the right the following transition rules are required.



Create New Gap

Fill old Gap

The General Technique.

To show that a set of transition rules is computation universal we will illustrate that we can embed a general purpose computation system in the array. That is, there is some configuration of states which can simulate the computer system. The direct approach is used. Since a computation system can be physically built out of wires, transistors, etc, it will be sufficient to show that there are configurations (initial assignments of states to the cells) that act like wires, etc., which can be interconnected to form the computation system. (This same approach was used by von Neumann and Codd.) In the cases of those cellular automata whose initial configuration is finite, we must also provide for extending the information capacity by an arbitrary amount.

[It was claimed by A. M. Turing and is now generally accepted that a particular very simple computer with an unlimited supply of tape to write on and read from can compute any computation that is effectively computable. This machine is called a Turing machine. It has been demonstrated that today's general purpose computers can

simulate the operation of the Turing machine if their memory capacity is unlimited. Thus we will only need to show the ability to simulate an extensible general purpose computer in the cellular array.]

A complete configuration will not be shown. We will only show that a sufficient set of basic components, such as the above mentioned wire, can be configured. In the case of the universal constructor, a moveable construction arm will also be needed.

Three basic elements are needed in most cases, the two-state case being the exception. These three elements are the fanout junction, the dead-end wire and the extensible tape. The junction is simply the intersection of a vertical wire with a horizontal wire. The junction will allow the duplicating or fanning-out of signals--i.e. a signal entering any leg of the junction will fan out the other three legs. The junction will also serve as a universal logic function. This requirement is met if either two or three simultaneous inputs (signals) yields no output while some other configuration of simultaneous inputs does yield an output. This is verified in each case.

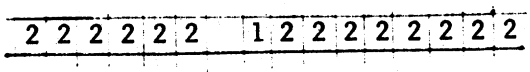
The dead-end is another basic element. By using the dead-end on two legs of the junction, the curve is obtained. The diode, or one-way gate, is created from basic elements as is the crossover. The logic element (analogous to the transistor), used in great numbers connected by wires, curves and crossovers now allows the configuration on the array of a large computation system. The extension of information capacity is discussed later.

The next chapter is an example of the application of these techniques.

IV. AN EARLY THREE-STATE UNIVERSAL COMPUTER

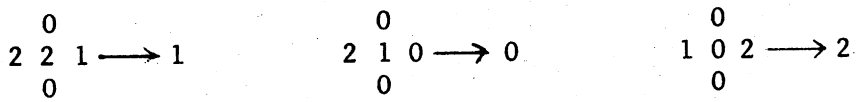
In this chapter it will be shown that a three-state, five neighbor cellular space is computation universal. Although it is shown in the next chapter that two states are sufficient, the three-state cellular space was found earlier and well illustrates the techniques employed.

The three states will be represented by the symbols 2, 1 and blank space or zero. The first of several needed useful elements is the wire. A special junction or fanout element and a dead-end for the wire will allow the attainment of our goal. The motivation for creating these elements is the desire to maintain an analogy to the physical wiring up of finite state machines with wires, transistors, etc. The need for a crossover results from our working in two dimensions.



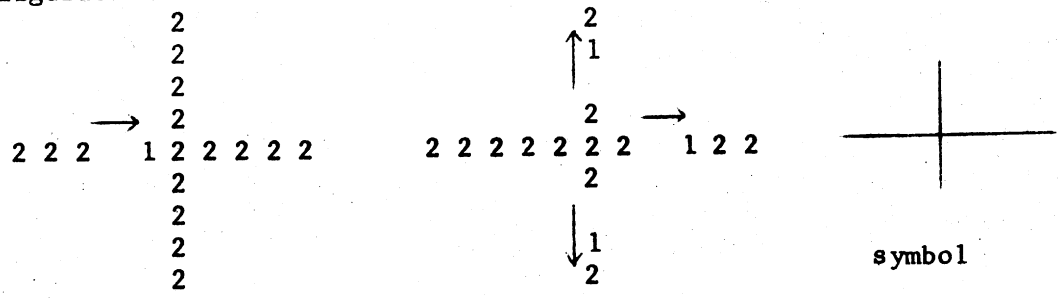
A Wire in a Three State Space

For the "10" signal to propagate to the right, the following transition rules are needed.



Propagation Transition Rules

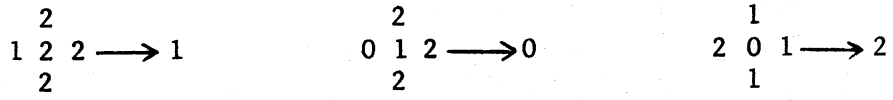
It will often be necessary to have a branching wire in order to send a signal to more than one place. The junction has the property that an entering signal will exit from the other three legs of the configuration.



time = t time = t+4 symbol

Fanning Out of Signal in the Junction Element

Three more rules must be added to obtain this action.

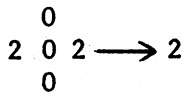


Fanout Transition Rules

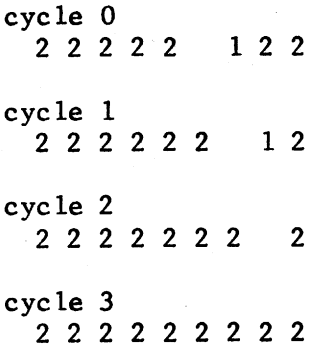
The reader should understand that as rules are added to create new elements, they must not conflict with previous rules--either stated transition rules or implied non-transition rules.

An interesting phenomenon in the search for universal cellular automata is the complicating effect of simple curves. In all cases the need for a simple curve was circumvented by the use of the previous fanout element with two of the output wires dead-ended.

The dead-end is simply a truncated wire. A signal reaching the end of the wire dies without changing the length of the wire. The dead-end requires the following rule to be added.



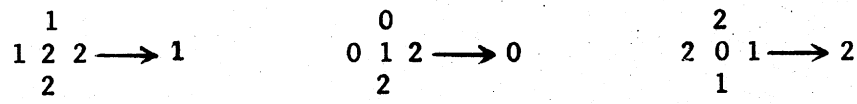
A Transition Rule Necessary for Operation of the Dead-end



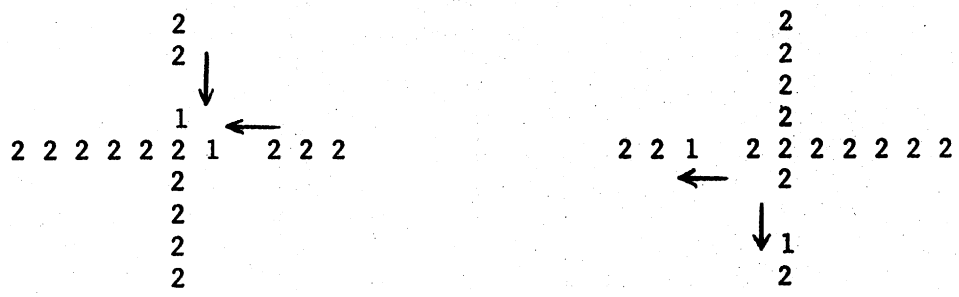
Simulation of the Dead-end Wire

Observing what happens to the junction element when three inputs arrive simultaneously, it is seen that the three inputs are mutually anihilated. This results from transition rules already written. What happens when two inputs arrive simultaneously can still be specified. The following right angled, two-input/two-output operation can be

achieved by adding three additional rules.

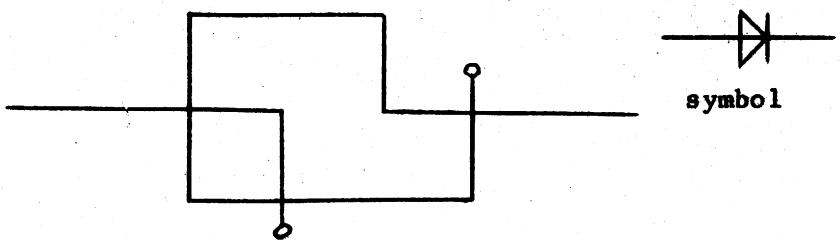


Rules Necessary to Cause Two-Input/Two-Output Operation



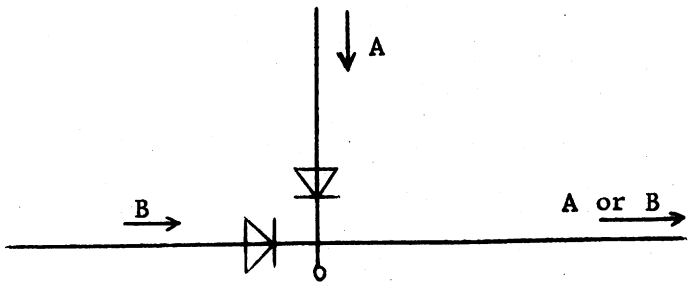
The Two-Input/Two-Output Element

If one had a diode (a one-way gate) then the above two-input/two-output would be an OR function. The diodes would be used to prevent signals from traveling out the inputs. This function is illustrated later. If wires are represented by lines, curves by right angles, and dead-ends by small circles, then the diode is configured as follows.



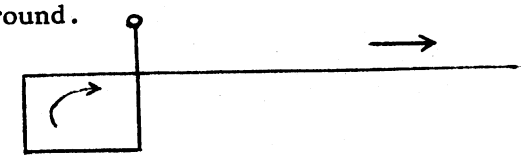
The Diode

The reason for the odd shape is explained by tracing through its operation. Consider first a signal entering from the right. It fans out three ways, one way being dead-ended and one further fanning out. Finally all three resulting signals simultaneously arrive at the left-most junction and are mutually anihilated as previously discussed. However, a signal entering from the left succeeds in getting through since at most two signals arrive at any junction at once. Now with the diode, the OR can be configured.



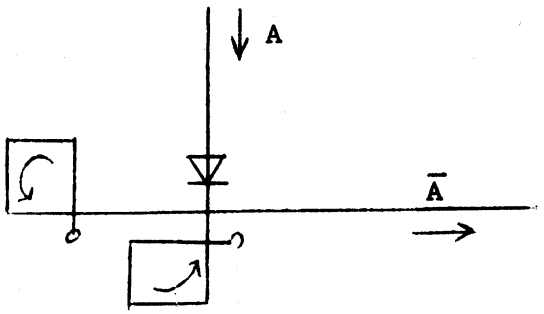
The OR Function

Another useful configuration is the clock or periodic emitter of signals. A signal circling in a loop will send a signal out the side wire each time around.



The Clock

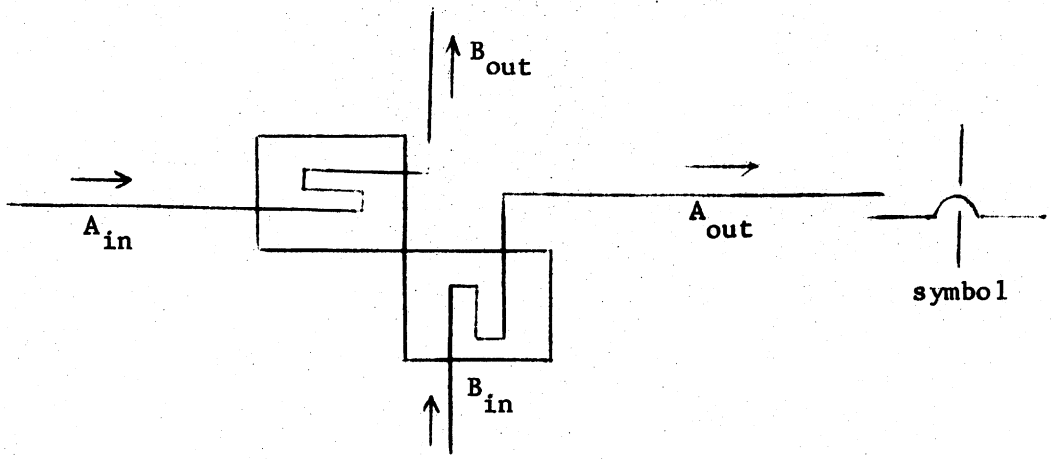
Now with two clocks and a diode, the NOT function (or invertor) can be configured.



The NOT Function

When a signal enters at A, there is no exiting signal due to the three input anihilation property of the junction, but in the absence of a signal the two clocks will generate an output, now due to the two-inputs/two-outputs property of the junction. Of course the clocks must be synchronized with the times allowable for signals to be at the junction.

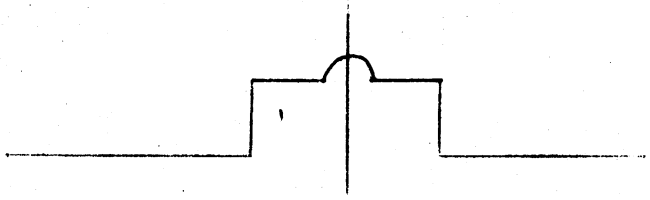
The OR and NOT functions form a universal pair of logic elements. There is only one remaining element, therefore, and that is the crossover. Then the logic can be "wired".



The Crossover

The purpose of the excessive curving inside each block of the crossover is to form a delay. Thus signal B entering from the bottom would fan out three ways. But the three paths to the A exit are equal forcing an anihilation of signals here due to the three simultaneous input anihilation property of the junction.

The crossing requires that the A and B inputs never arrive simultaneously. If one assumes a minimum signal spacing of sufficient size, then the following configuration can always be used to assure non-simultaneity of inputs.



Avoiding Simultaneous Arrivals

Our goal has been to be able to configure in a finite space any finite state machine with these elements. In particular, since we can transmit a binary signal (presence of signal indicates one state, absence of signal indicates other state) from any area of space to any

other, and since signals can be combined by a universal logic function such as the NOR (meaning neither input present), it is clear that all the components necessary to construct any general purpose computer are present within configurations of states in the array. A simple way to show this universality is to show that a one-dimensional array of n-state machines can be configured for any finite value of n. When n is suitably large, a linear array can simulate the Turing machine. This is possible, for example, if each of the simulated n-state machines represents a square of the Turing machine's tape, but also containing the finite-state machine part of the Turing machine.

Alternately, the finite eight-state cells of Codd or the twenty-nine-state cells of von Neumann could be simulated. Then an array of simulated Codd cells could perform as described by him.

Several other infinitely configured three-state universal cellular spaces were discovered. Common elements include a universal logic element, a wire, a dead-end and a crossover. Some had fewer transition rules. Different diode and crossover configurations, of course, had to be obtained since the properties of the junction were not identical to the case described here.

V. THE INFINITELY CONFIGURED TWO-STATE UNIVERSAL CELLULAR SPACE

One of several methods of illustrating universality is to show that two-state cells in the proper configuration can simulate any n-state cell--in particular the twenty-nine state von Neumann or eight-state Codd cells, or the three-state cells of the previous chapter. However, following a similar approach as with the three-state cellular space, it will be sufficient to show the ability to arbitrarily connect universal logic elements to create arbitrary finite-state machines. Then an infinite one-dimensional array of these finite-state machines can simulate any Turing machine (by the same argument as before) and is therefore universal.

If the two states are represented by 1 and blank space (or 0), three transition rules can be written to define this cellular space.

$$\begin{array}{c} 1 \\ 0\ 1\ 1 \end{array} \longrightarrow 0 \qquad \begin{array}{c} 0 \\ 1\ 0\ 1 \\ 1 \end{array} \longrightarrow 1 \qquad \begin{array}{c} 1 \\ 1\ 0\ 1 \\ 1 \end{array} \longrightarrow 1$$

Rules for the Two-State Computation Universal Cellular Space

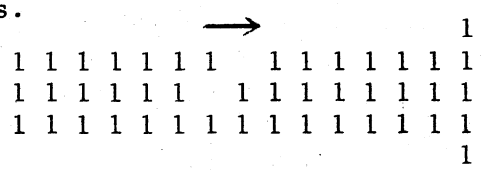
The first of these rules requires corners to disappear. The other two assure that gaps (zeroes) surrounded by three or four ones will be filled in. In the following illustration of a signal propagating along the wire, the zero or quiescent state is also denoted by the blank. $\longrightarrow$

```
1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Wire and Signal

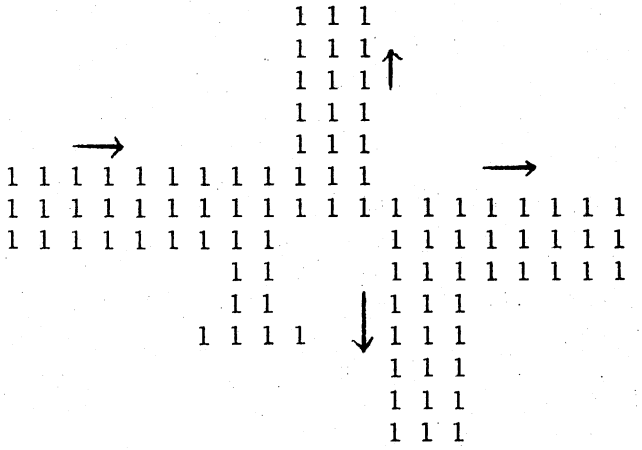
Note that the signal travels on one side of the wire. The wire will be symbolized by a straight line with an arrow used to indicate the side of the wire on which the signal is traveling. It should be clearly understood that the blank area above the wire represents cells in the zero state and not the 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 along truncated wires.



The Dead-end

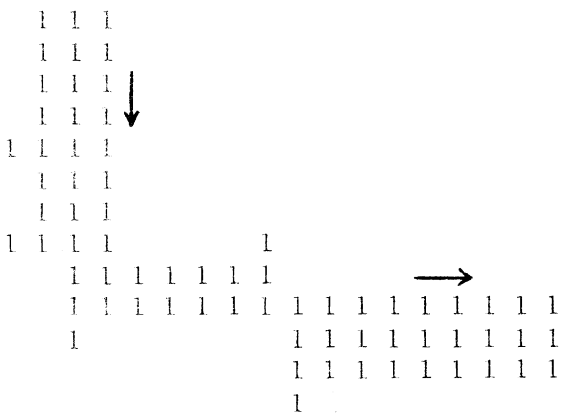
The junction or fan-out is used to create new signals. A signal entering the fan-out from the input side leaves from the other three arms. This element will also be used with the above dead-end to produce the curve.



The Junction

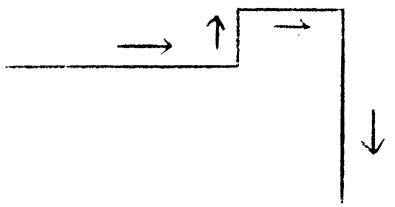
This junction is not bilateral. A signal entering any of the side outputs dies. Notice that by using two dead-ends on two outputs, the diode (one-way gate) is obtained. However the diode is not needed in this cellular space.

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. Two of these curves can be obtained directly from the fanout with dead-ends on two output wires. The inside-to-inside curve must be synthesized.



The Inside-to-Inside Curve

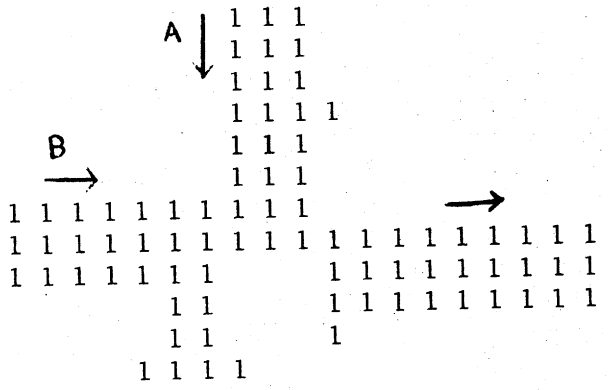
The outside-to-outside curve can now be made from the above curves.



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".

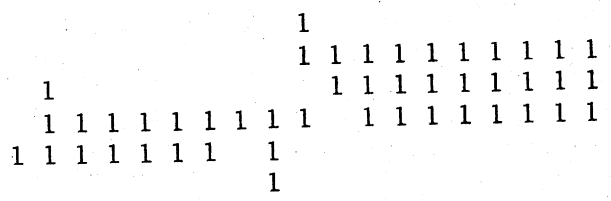


TRUTH TABLE

B \ A	0	1
0	0	0
1	1	0

The Logic Element

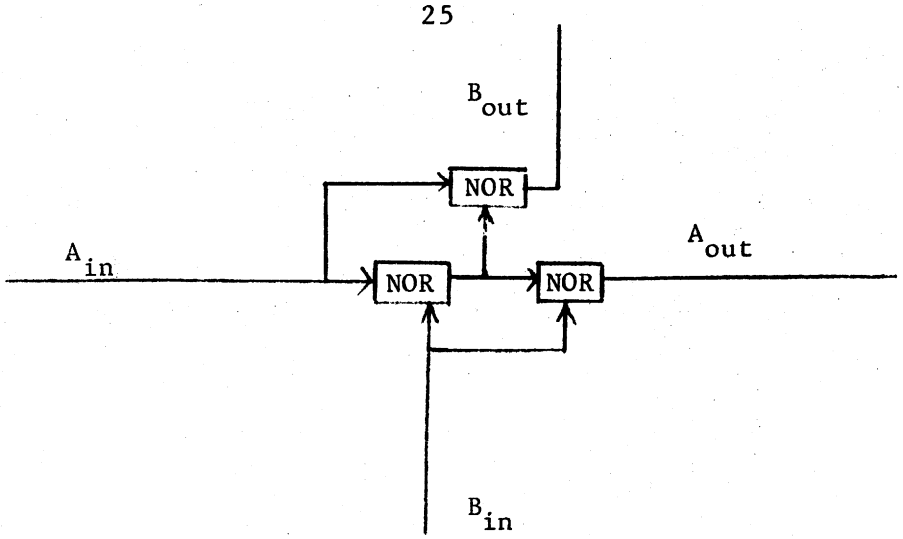
If the B input is from a clock (i.e. a periodic emitter of signals) then this logic element becomes a NOT function. Thus a clock is needed.



The Clock

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

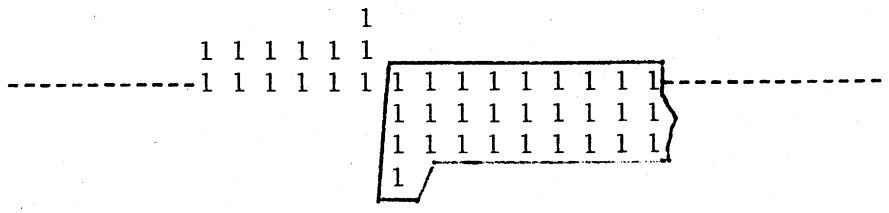
The operation of the logic element requires that the signals be synchronized. The clock defines an interaction time and all wires are constructed to maintain the synchronization of signals. The construction of the crossover is achieved by the following logical configuration. However it is required that only one signal arrive at once. By moving the location of the crossover slightly, as in the three state automaton of the previous chapter, non-simultaneous arrivals can be assured. The NOR function used in this diagram can be constructed from the logic element with a NOT function feeding into the B input.



The Crossover

A much more efficient crossover can be constructed.

The extension to three dimensions and seven neighbors (six face-neighbors plus the cell itself) is straight-forward. The same three rules apply assuming the other two neighbors are in the zero state. The above automaton can exist in a plane of this space, 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. Subsequent turns will now be out of the plane.

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 NOR function. (Minsky 1967) Thus we can wire up any function. (Delays can be built with a few extra curves.) Specifically, the twenty-nine

state von Neumann or eight-state Codd cells could be simulated achieving both computation and construction universality except that an infinite number of simulated cells must exist, i.e. it takes several of these cells to simulate one of the Codd cells even if it is merely in the zero or quiescent state.

VI. THE THREE-STATE FINITE UNIVERSAL COMPUTER

The undesirable requirement of an initial configuration involving an infinite number of cells can be eliminated by adding another state. (If it is assumed that the cellular computer is not allowed to continue growing without bound after the computation is complete, then three is the minimum number of states for this behavior. The reason for this is that the ability to grow more memory with two-state cells, as required by a finite initial configuration, necessitates the following transition rule.

$$\begin{matrix} & 0 & \\ 1 & 0 & 0 \end{matrix} \longrightarrow 1$$

But this rule will cause an unboundable propagation from some edges of the automaton. Thus this and the previous automaton are each minimal for their respective types of behavior.)

Since a computation may require an arbitrary amount of memory space, some method must exist for increasing the information storage by arbitrarily large amounts. The method is to use an arbitrarily extensible special wire or tape 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 two register machine (Minsky 1967). 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 two operations 1. add one to a register and 2. subtract one from a register and if the result is zero, branch to a specified operation. Since addition and subtraction are operations that require only a finite state machine to carry them out (as opposed to multiplication), only the ability to add and subtract one 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 one, a signal is sent out 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.

The Appendix contains the computer simulations of these elements and a listing of the transition rules. The states are 2, 1, and the quiescent state 0 or blank space.

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

```
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2   1 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Wire (Propagation to the Right)

The following junction construction has the properties 1. a single signal entering any 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 the arrival of a signal or signals.

```
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
```

The Junction

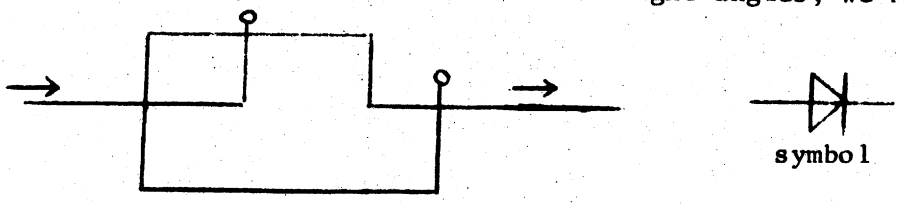
The dead-end is simply a truncated wire. However, if an extra 2 is placed at the end of the dead-end wire as shown, then the signal

```
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2   1 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

The Special Extensible Wire

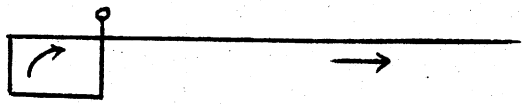
will reflect from the right end (above) with the special wire being lengthened by two cell edge lengths.

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. Representing wires by lines, dead-ends by circles and curves as right angles, we have:



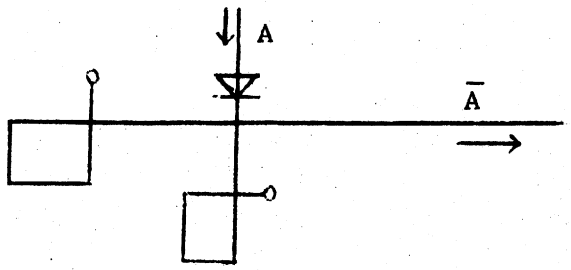
The Diode (Signal Will Pass Only from Left to Right)

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



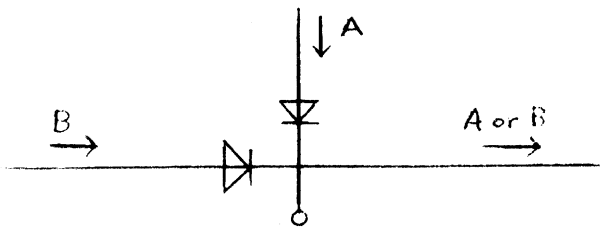
The Clock

The diode and the clock can be used to make a NOT function.



The NOT Function

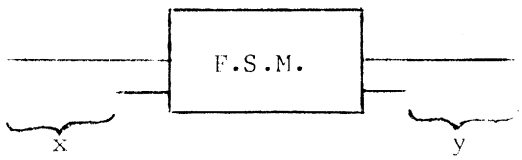
The OR function is just a junction with diodes on the inputs.



The OR Function

The NOR function can be obtained from the OR with its output fed into the input of a NOT. The NOR can be used as in the two-state case to form a crossover.

The reception of echo signals, the program logic necessary to build a finite-state machine, etc., are now achieved by straight forward constructions. Thus the finitely configured three-state universal cellular automaton can be built.



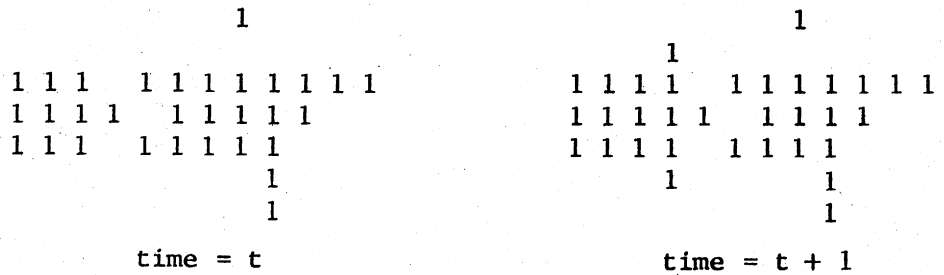
The Two Register Machine

VII. THE FINITE UNIVERSAL COMPUTER WITH NINE-NEIGHBOR CELLS

An alternate approach to achieving universality with a finite initial configuration is to increase the neighborhood size over the earlier five-neighbor cells. Including the four corner-neighbors of a cell is perhaps the simplest way to do this. Thus this chapter is going to describe a finite universal computer of two-state, nine-neighbor cells.

There appears to be another universal array of two-state, nine-neighbor cells, besides the one described in this chapter. William Gosper and others of Project MAC at MIT have recently demonstrated a large number of behaviors achievable by configurations of Conway's "Life" cells. It is believed that this array is also universal, but the description is lengthy. In this automaton, the basic signal is a self-propagating "glider". These results will probably be documented by him at a later date.

In parallel with the description of the previous chapter, the automaton of this chapter will employ three basic components: the dead-end, the junction and the special extensible wire. Computer simulations of these components and a list of the transition rules appear in the Appendix. The approach will be identical to that in the previous chapter--Minsky's two register machine is to be simulated with two extensible wires representing each register.



Wire and Signal and Dead-end

The signal is shown at two instances of time since its form alternates between these two forms. (This introduces a new problem--the consideration of the phase of a signal. A fourth element, the phase

convertor, will insure the correct phase of the signal for the desired operations. Any signal passing through the phase convertor will exit with a phase determined by the location of the convertor regardless of entering phase. (The two extra 1's merely strip the signal's 1's from it as it passes. When the signal is out of range of these 1's, the signal can reform its 1's.)

```
          1

  1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1

          1
```

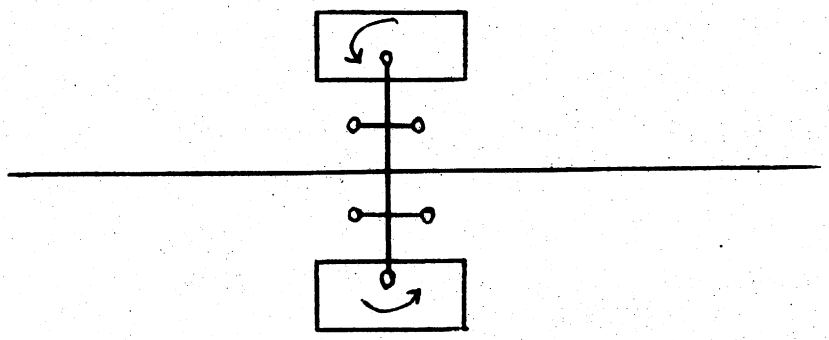
The Phase Convertor

The junction has the following properties: 1 one input will fan out the other three wires, 2 two inputs not at right angles are mutually anihilated, and 3 three simultaneous inputs are also anihilated.

```
            1 1 1
            1 1 1
        1   1 1 1   1
            1 1 1
  1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1
            1 1 1
        1   1 1 1   1
            1 1 1
            1 1 1
```

The Junction

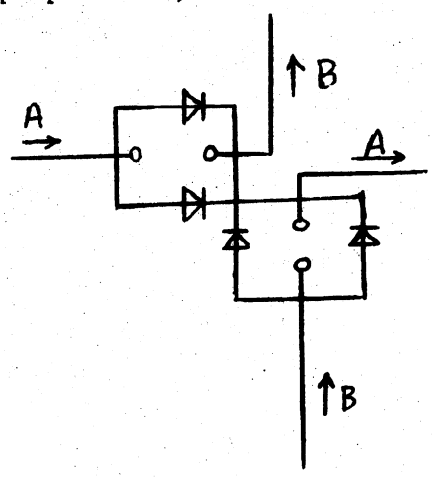
The clock, as previously, is simply a signal circling in a loop with an exit. Using two clocks and three other junctions as in the following figure, the diode can be obtained.



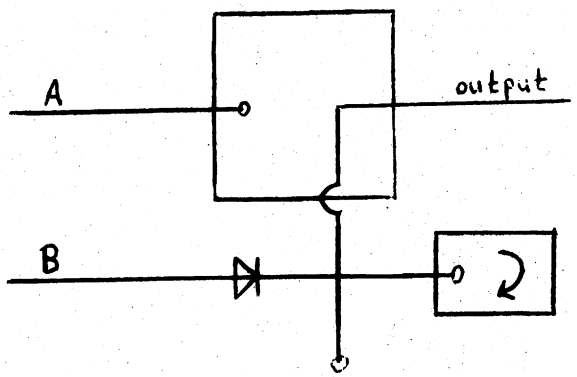
The Diode

How can this symmetrical component allow passage of signals in only one direction? It can due to the synchronous operation of all these cellular automata. The clocks are synchronized so that a signal entering from the right arrives at the center junction simultaneously with the two clock signals, thus being anihilated. A signal entering from the left, however, fans out with one signal getting through and the two others dying at the other two junctions by the two-input anihilation property.

The crossover and NOR logic element are shown. (The reader is urged to verify the operation of these components from the junction properties.)



The Crossover



The NOR Function

The remaining component is the extensible wire. The extra one at the right end of this segment causes a signal to "bounce" off the end, generating an echo and lengthening the wire.

```
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

The Extensible Wire

The design of this automaton now corresponds to that of the previous chapter.

* John H. Conway has also shown the Life cells to be universal independently of William Gosper's work except that he used the "glider gun" (a configuration that periodically emits a self-propagating signal) discovered by Gosper. Future issues of the Scientific American magazine will have a description of both Gosper's and Conway's works as well as a brief mention of the author's work. Neither Gosper nor Conway have published their work as far as is known by the author.

VIII. THE SELF-REPRODUCING UNIVERSAL COMPUTER, UNIVERSAL CONSTRUCTOR

The four-state universal computer, universal constructor is considerably more complex than the previous three automata. Not only must this automaton be capable of computing any (computable) computation but it must also 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 a copy of itself. (For a discussion of how such offspring can evolve and improve, see von Neumann 1966.)

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, since we plan to have the automaton construct a passive configuration. 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 are given in the Appendix. The states are represented by blank (and 0), 1, 2, and the letter x.

Let us begin with the wire and signal, and the dead-end.

```
x x x x x x x x x x x x x
        2 1             x
x x x x x x x x x x x x x
```

The Wire and Signal and the Dead-end

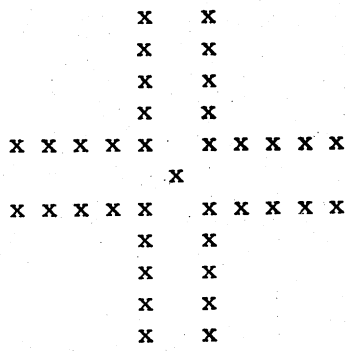
The wire and all other components are constructed from state x. Constructions consisting of only state x are very stable, the only transition being the following.

```
    x
x   0   x  ⟶  1
    x
```

None of the components, while passive, will contain this local configuration. Therefore, any construction consisting of the

components to be described, will remain passive until activated.

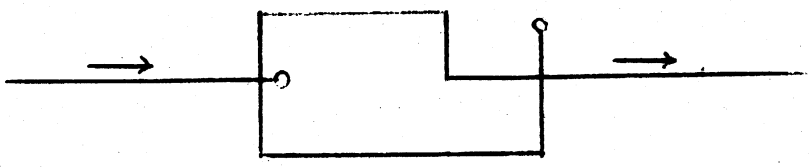
The junction has the same properties as had the three-state automaton, namely 1. a single entering signal fans out the other three legs, 2. two inputs at a right angle give two outputs on the other two wires, and 3. three simultaneous input signals are mutually anihilated.



The Junction

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. Also if two signals arrive oppos ite each other, they will be mutually anihilated, a property not possessed by the three state automaton. These properties are verified in the computer simulations.

The curve is obtained from the junction with dead-ends on two of the output legs. The diode is obtained as previously, or by the following alternate configuration.

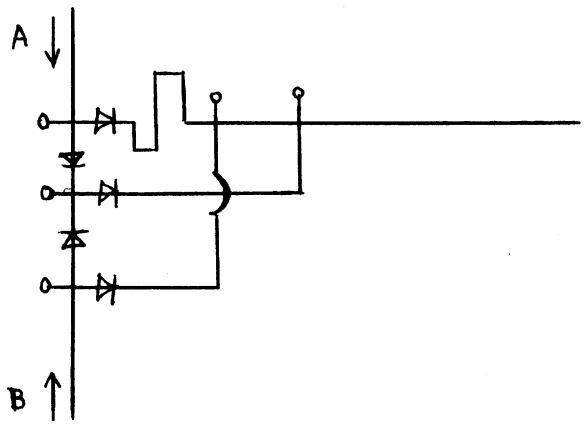


The Diode

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

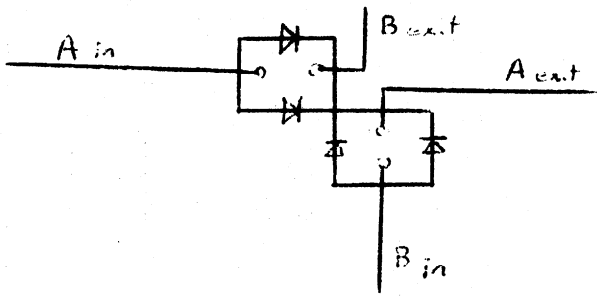
Errata

page 36, third line from bottom: "Similarly the clock, NOT function and NOR function can be constructed." This sentence is not exactly correct. True, the clock and NOT can be constructec as in chapter 6. However, due to the one unit delay of the junction when only a single input is present, the OR as shown there will not work. Although this can be corrected by another configuration, we will correct it by adding two more transition rules, namely (12x2x2) and (212001). The new rules cause the two-input/two-output property of the junction to hold even if one of the inputs is delayed by one time unit. All other components operate as before. The new OR function is shown below. Its operation can be verified by tracing through its operation and remembering that curves (junctions with two dead-ends) also have a delay of one unit.



The OR Function

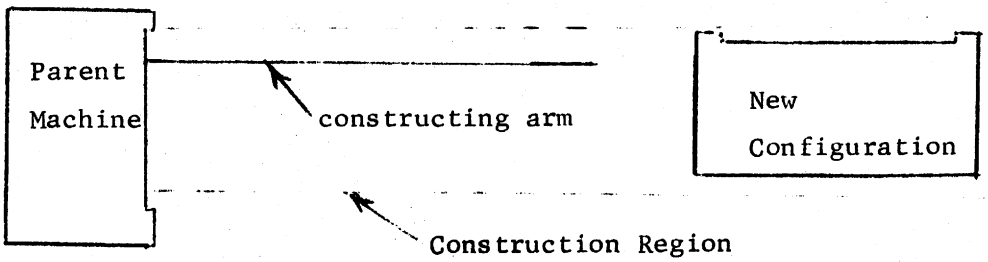
for this unacceptability is that we only want to construct a passive configuration and to use a single activation signal to introduce a signal into all wires and clocks to "start" the newly constructed machine. With a passive crossover this problem is avoided.



The Passive Crossover

A signal entering at B above would not be able to exit at the A exit due to the above mentioned property of the junction, namely that two signals entering opposite each other cancel.

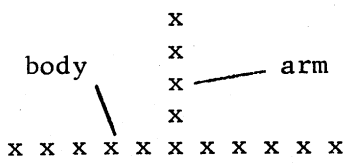
The logic operations of this automaton are now completed. The operation of the arm is illustrated now.



Construction

The horizontal arm can move vertically within a fixed range, but can extend arbitrarily far into a construction region. The construction of automata which do not fit within this space (e.g. self-reproducing automata in the same orientation as the parent) 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. The arm is attached to another row of cells called the body.

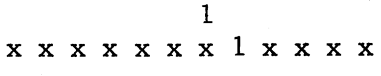


Attachment of Arm to Body

The operation of construction proceeds as follows:

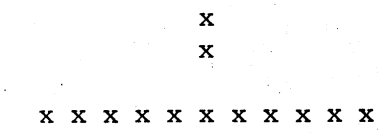
1. Two special signals (called single wings) sent from each end of the body collide at a location determined by the timing of signal origination.
2. The collosion produces an arm bud.
3. Further pairs of single wings colliding at this location cause the arm bud to grow longer.
4. Once the arm has reached 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 wing propagates out the wire, destroying the wire as it goes--i.e. it leaves 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 position of 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 signal is illustrated.



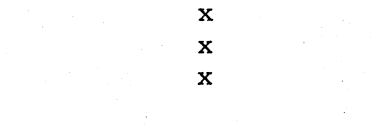
Single Wing Signal

Two meeting single wings leave the following arm bud.



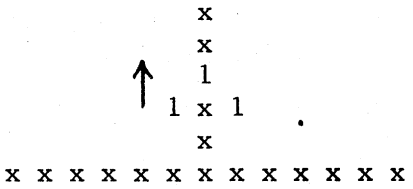
Arm Bud

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



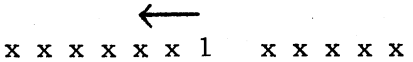
Arm with Gap

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



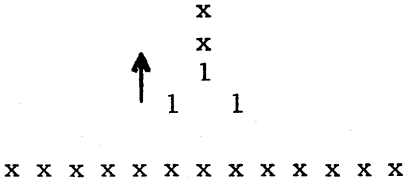
Double Wing Signal

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



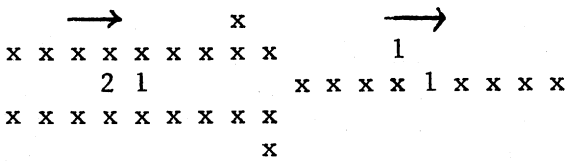
Echo Signal

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 different cases of the erasing wing reaching the end of the wire are illustrated in the computer simulations of the Appendix. By constructing the new automaton column by column and top to bottom in a raster pattern, the arm will need to consider only local configurations in which there are no x's already there (at the end of the arm) 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. (See the Appendix for further detail.) In particular, the arm will not have to fill an x between two existing x's.

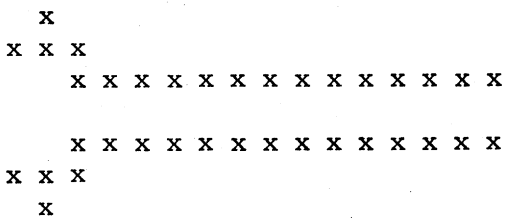
So far we have not described how the single wing signals are generated. The following configuration will transform all entering logic type signals into single wing signals.



Single Wings Being Generated

Logic type signals entering from the left will exit from the right as single wing signals.

Activation is achieved by the activation mechanism.

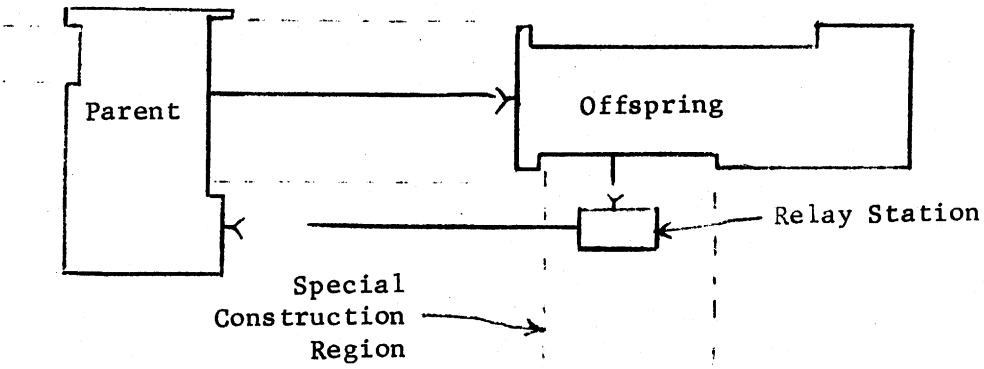


Activation Configuration

An erasing wing entering the above element would generate a "12" type logic signal. This activation signal would now travel by fanouts and crossovers to every place where it is desired to have a wire or clock initialized with a signal. After an erasing wing has entered, it is left in the following condition and must be "cleaned up" for subsequent use. Of course, if no further communica-

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

Now that by cleaning up the activation mechanism, communication is possible from the parent to the offspring. To allow information to flow in the opposite direction, the following special construction region is added to the offspring so that it can construct the "relay station".

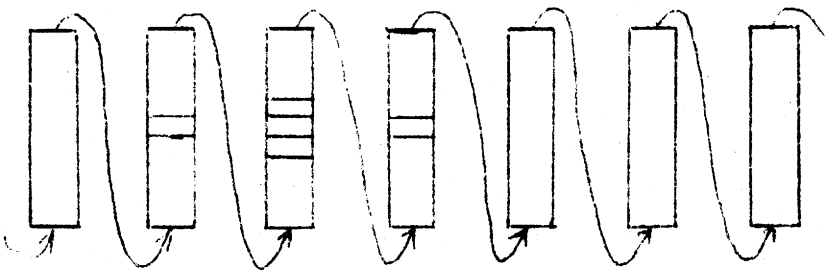


Two-Way Communication

Banks (1970) describes another four state universal constructor that has only a one-shot activation mechanism--it is not reusuable. This paper describes how sufficient information can be communicated in this case. It also illustrates how a permanent two-way wire can be constructed to connect the parent with offspring.

IX. UNIVERSAL ONE DIMENSIONAL CELLULAR SPACE

The two-state universal cellular automaton of chapter V can be constructed in a finite width of the two-dimensional space. By considering an infinite one dimensional tape made up of slices of the two state machine configured in two dimensions (see diagram), as pointed out by Fredkin, 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. Thus each cell has the same five neighbors as in the two dimensional space and operation can be identical.



Forming a One Dimensional Space from the Two Dimensional Space

X. DISCUSSION

Starting and Halting.

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

In the same sense that the input to a universal Turing machine is initially configured on its 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.

The Continuous Space.

The location of any cell of dimensions dx and dy can be measured by the coordinates x and y. If S(x,y,t) represents the state of this cell at time t, it can take on values 0 and 1 for the universal two-state cells of chapter V. The value of S(x,y,t+dt), where dt is the duration of the time steps, thus depends on S(x,y,t), S(x+dx,y,t), S(x-dx,y,t), S(x,y+dy,t), and S(x,y-dy,t) according to the set of transition rules. Following normal practice, spatial second derivatives can be defined.

$$S_{xx} = \frac{\partial^2 S(x,y,t)}{\partial x^2} = \lim_{dx \to 0} \frac{S(x+dx,y,t) + S(x-dx,y,t) - 2\,S(x,y,t)}{dx^2}$$

and similarly for S_{yy}. The two values S_{xx} and S_{yy} can take on values 0, +1, -1, +2, and -2. Interestingly, S(x,y,t+dt) can be expressed as a function of only S, S_{xx}, and S_{yy}, if the transition rules of the two-state space are applied. Defining this function as F and letting dt approach zero, one gets:

$$\frac{\partial S}{\partial t} = F\,[\,S,\, S_{xx},\, S_{yy}\,]$$

or explicitly for these transition rules as:

$$\frac{\partial S}{\partial t} = \begin{cases} 1 \text{ if } S = 0 \text{ and } S_{xx} \cdot S_{yy} = 2 \text{ or } 4 \\ -1 \text{ if } S = 1 \text{ and } S_{xx} = S_{yy} = -1 \\ 0 \text{ otherwise} \end{cases}$$

Some readers may recognize the above function form as a type of wave equation--i.e. a second order differential equation with first order time differentiation. This result is more interesting in three dimensions. (Recall from chapter V that the two-state space is still universal in three dimensions.) It is still possible to express the future state of a cell as a function of only second derivatives and of its own current state, hence:

$$\frac{\partial S}{\partial t} = G [S, S_{xx}, S_{yy}, S_{zz}]$$

although the explicit form is now more complex. Now if S is allowed to vary continuously, rather than discretely, this equation defines a sort of universal continuous cellular space.

Other Applications of Cellular Automata Theory.

Smith (1970) and others are interested in applying cellular automata theory to biological phenomena. The study of DNA interactions, consisting supposedly of a set of four "symbols" inside the spiral helix, is one interesting area.

Another possibility is the design of electrical circuits with self-wiring capability. An LSI 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 perhaps be a complete parallel 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, more complex, and more variable circuitry could be used. Shoup (1970) 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.

XI. SUGGESTIONS FOR FURTHER STUDY

The following list of questions and comments can be interpreted as suggestions for further investigation.

1. Are there other two-state, five-neighbor infinitely configured universal cellular automata? If we restrict ourselves to isotropic cells (as defined in the Glossary) then there are only twelve transition rules that can be written, one of these being specified (the quiescent non-transition requirement). Thus there are only 2^{11} automata in this class in all.

One other has been found. If the present automaton is embedded in a region of cells in state zero which is in turn embedded in an infinite region of state 1 cells, then the same machine can be considered since state 1 matches the quiescent rule form. Notice that this is not just swapping the names of the states.

2. The two-state five-neighbor, three-state five-neighbor, and two-state nine-neighbor cells are minimal in the number of states required for achieving their respective types of universality. Is the four-state universal constructor also minimal?

3. Are there much simpler spaces in the sense of having many fewer transition rules?

4. How large are the total configurations? The author has not yet seen it worthwhile to construct a complete, detailed computer. Such an effort would require a large amount of resources to show something already known.

5. Is there a simple cellular model of physics?

6. Is there a line between the automatic type of self-replicating cells of Fredkin and Winograd (See chapter II) and those of the universal construction type of self-reproduction? (Minsky)

7. What are the characteristics of configurations on different shaped cells such as hexagons, triangles, spheres, etc., or in higher dimension?

8. How much more powerful are cells with asymmetric transition rules? The answer to this question could be quite useful to the goal of building practical cellular automata on integrated circuits. For instance, information can now be densely packed in wires since the length of signal can now be only one cell.

9. How complex must the cells be in order for a universal constructor to also be able to erase a configuration? If the configuration is not actively defending itself, it seems that the cells of this thesis meet this requirement since a 1 introduced into the edge of a wire tends to propagate by devouring the wire. This action should be verified for all types of local configuration that could be encountered.

10. What other useful measures of complexity of behavior are there other than universal computation (finite and infinite initial configurations) and universal construction?

SUMMARY

Table of Results.

This table is drawn for infinite two-dimensional arrays of square cells. It gives results, discoverer and other information.

Neighbors per Cell

States per Cell	5	9	85
2	Universal Computer Infinite initial configuration req'd [author]	Universal Computer Finite initial configuration. [author; also Wm. Gosper of MIT has shown Conway's Life cells are universal]	Universal Constructor [Codd] The cells were shown capable of simulating Codd's 8-state cells.
3	Universal Computer Finite initial configuration is sufficient, [author]		
4	Universal Constructor [author]		
8	Universal Constructor This was the first reduction of von Neumann's cells. [Codd]		
13	Universal Computer These cells req'd only two rows of the array. [Smith]		
29	Universal Constr'tr [von Neumann] This was the first cellular automaton shown to be universal		

GLOSSARY

Activation, the process by which a universal constructor causes a new passive construction to become active or operative, by the introduction of a signal.

Automaton, a finite or infinite-state machine. A property is the discreteness (versus continuous) of values for logic levels, etc.

Cell, a finite-state automaton in an iterative array. Every cell has a set of neighboring cells with which it communicates. At discrete moments of time the cell assumes a state, this new state depending on the old states of the cell and its neighboring cells according to a set of prescribed state-transition rules.

Cellular Automaton, an array, usually infinite and two-dimensional, of identical interconnected cells which simultaneously undergo state transitions at discrete moments of time (see Cell).

Cellular Space, refers to the entire collection of cells of a cellular automaton; it is defined by a set of transition rules, the shape of the cells (e.g. square), the dimensionality, and the neighborhood function (how the cells are interconnected).

Computation Universal Cellular Space, a cellular space capable of supporting a universal computer, i.e. there are initial configurations of states that can perform any computation. (see Universal Computer)

Configuration, an assignment of states to the cells of an area of the array at some time.

Construction, the process by which a universal constructor creates a new, disjoint configuration by growing an arm into an area of quiescent cells and changing the states of these cells to the desired configuration.

Construction Universal Cellular Space, a cellular space capable of supporting a universal constructor. It must also be a computation universal cellular space. (see Universal Constructor)

Finite Universal Computer, a universal computer which can begin operation with only a finite non-quiescent initial configuration, i.e. only a finite region of the infinite array has cells in the non-zero or non-quiescent state.

Garden-of-Eden Configuration, a configuration with no preceeding configuration which would yield it. A Garden-of-Eden configuration can therefore exist only in an initial configuration. Practically all interesting cellular spaces have Garden-of-Eden configurations.

Initial Configuration, the initial assignment of states to the cells of an area of the array.

Isotropic Cellular Space, a cellular space with symmetrical transition rules. If completely isotropic there is no preferred direction. Also a mirror image configuration will always maintain its reflection symmetry with the real configuration.

Iterative Array, a regular arrangement of cells such that any region of the array looks exactly like every other except for the states of the cells in the region, and except for boundary regions of the array if it is finite in extent.

Neighbor Cell, a cell which contributes its state information directly to another cell, thereby influencing the next state of that cell. Usually a cell is considered as one of its own neighbors.

Neighborhood, the set of neighboring cells on which the next state transition of a cell may depend. All cells, if identical, have the same geometrical shape of their neighborhoods.

Quiescent Cell, a cell in the quiescent state. (see Quiescent State)

Quiescent Space, a region of the cellular array consisting of quiescent cells.

Quiescent State, a distinguished state, denoted by 0 or blank, which obeys the following transition rule.

$$\begin{array}{ccc} & 0 & \\ 0 & 0 & 0 \longrightarrow 0 \\ & 0 & \end{array}$$

Passive Configuration, a configuration which remains unchanged during a time step.

Self-reproducing Automaton, a cellular automaton which can support an initial configuration which has the property that after a finite time there are two or more configurations identical to the original configuration.

Transition Rules, the set of rules specifying the new state of a cell as a function of the present state of the cell's neighbors. A typical transition rule has the form

$$\begin{array}{ccccc} & & N & & \\ W & C & E & \longrightarrow & R \\ & & S & & \end{array}$$

where the letters stand for Current state, Result state, North, East, South and West neighbor's states.

Universal Computer, a cellular array whose configurations at different times may be interpreted as performing any computation.

Universal Constructor, a configuration capable of constructing into quiescent space, another configuration which may be of arbitrary size and may itself be capable of universal computation and universal construction. (The existence of Garden-of-Eden configurations may preclude the construction of arbitrary configurations; hence this definition.)

APPENDICES

These appendices are reproduced directly from the computer simulation printout except for reformating so that each cell appears to have equal height and length, since the computer printout did not allow equal vertical and horizontal spacing.

Appendix I contains the simulations of the components of the universal computer configured in an array of two-state, five-neighbor cells.

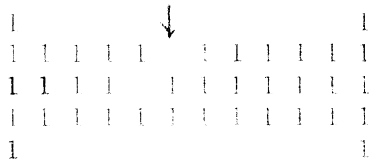
Appendices II, III and IV contain rule listings and simulations of the components of the three-state finitely configured universal computer, the two-state nine-neighbor finitely configured universal computer and the four-state universal constructor, respectively.

The computer simulation program was written to run on the Multics time-sharing system and is available on file at Project MAC. Another version of the program was written in LISP to run on the ITS time-sharing system. (Actually such simulation programs are easy to write.)

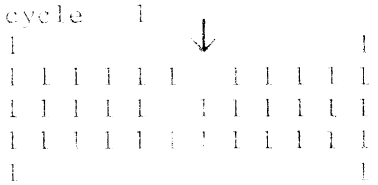
APPENDIX I: THE TWO-STATE INFINITELY CONFIGURED UNIVERSAL COMPUTER

Computer Simulation of the Wire and Dead-end.

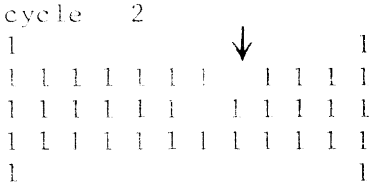
cycle 0



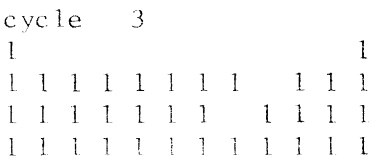
cycle 1



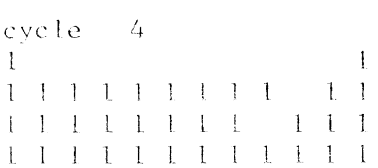
cycle 2



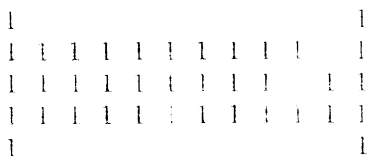
cycle 3



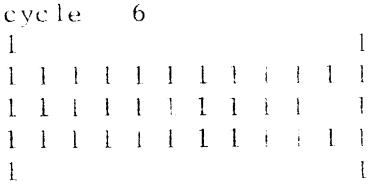
cycle 4



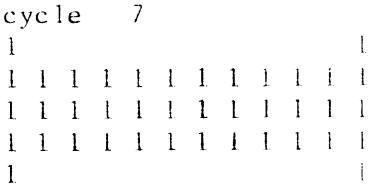
cycle 5



cycle 6



cycle 7



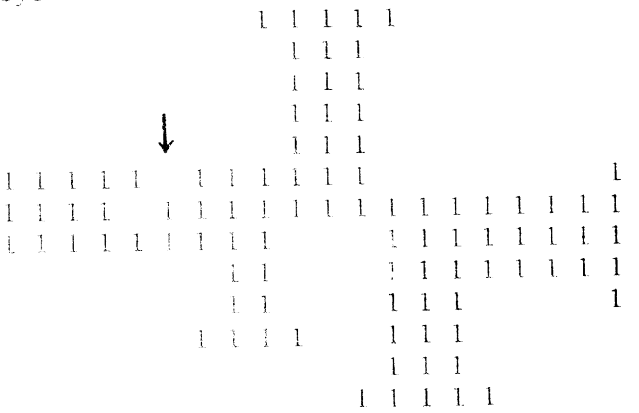
END

Notice particularly the disappearance of the signal in cycles 6 and 7

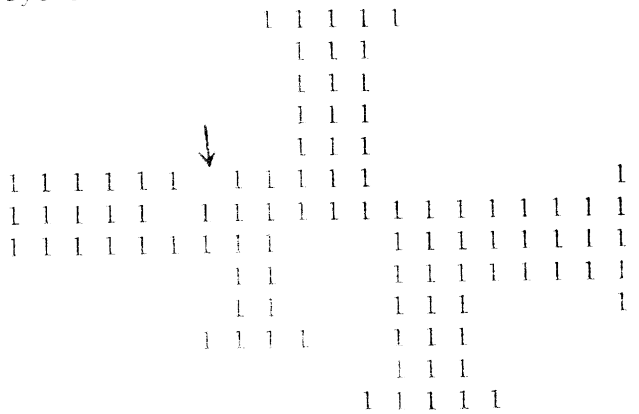
COMPUTER SIMULATION OF THE FANOUT.

The signal entering from the left will fanout the other three arms of this element. Dead-ends are used to kill these signals in this simulation--normally they would proceed to more useful elements.

cycle 0

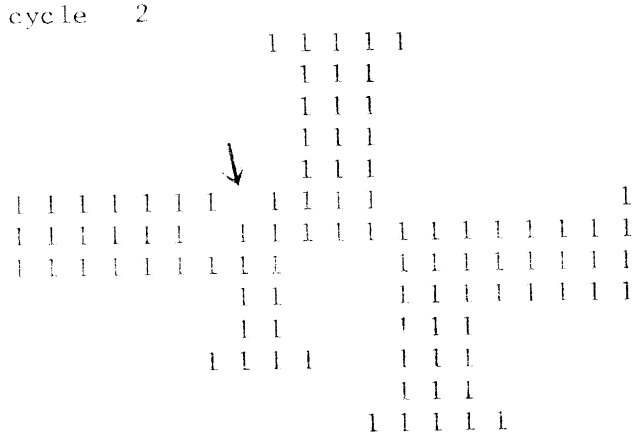


cycle 1



In these three cycles, the signal is entering the fanout.

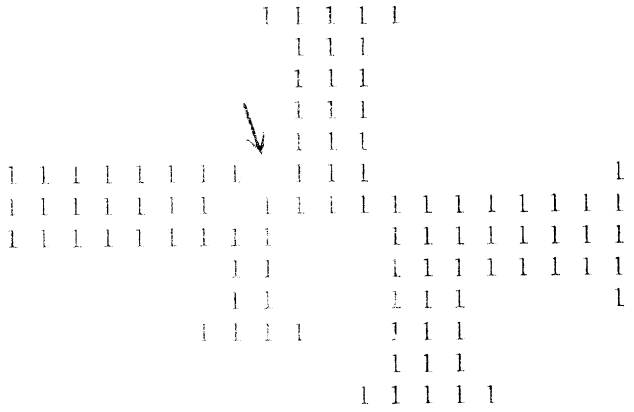
cycle 2



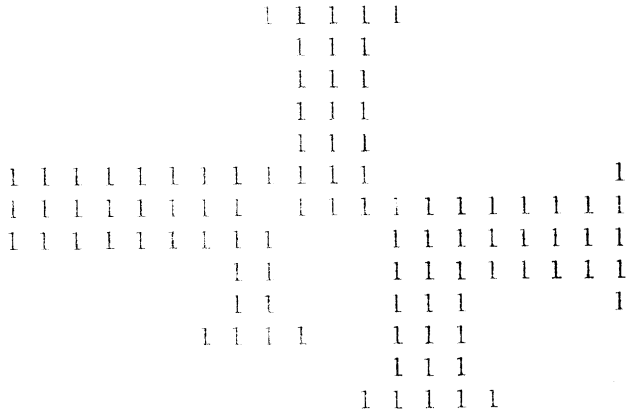
(continued)

These next three cycles of simulation show the signal moving to the underside of the junction.

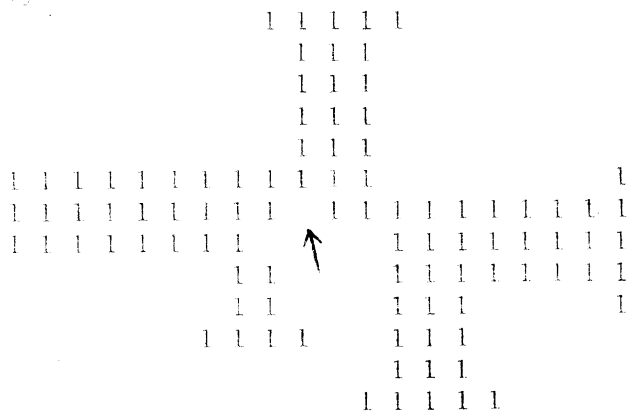
cycle 3



cycle 4



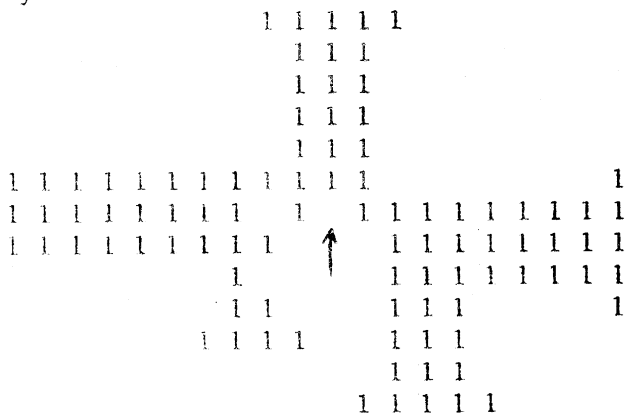
cycle 5



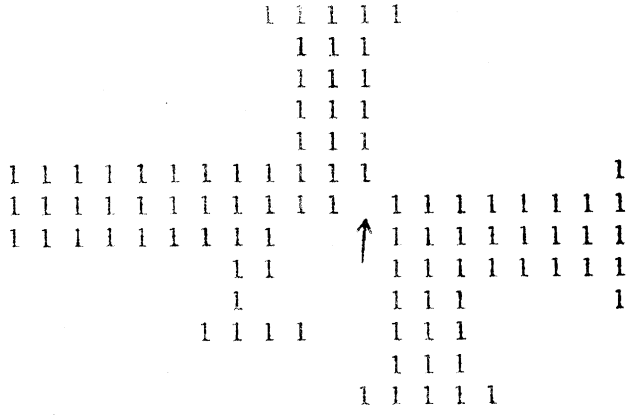
(continued)

In cycle 8 the signal is just beginning to start out the exit wires.

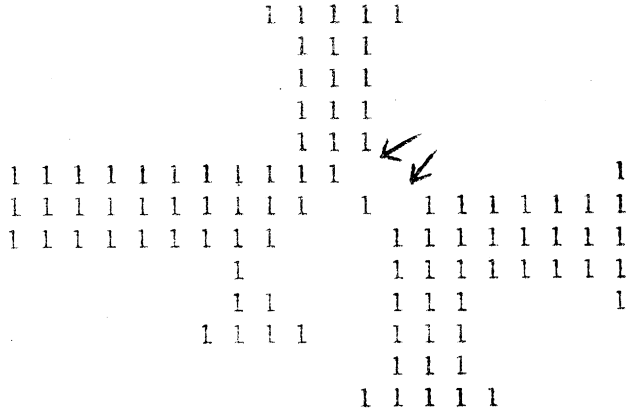
cycle 6



cycle 7



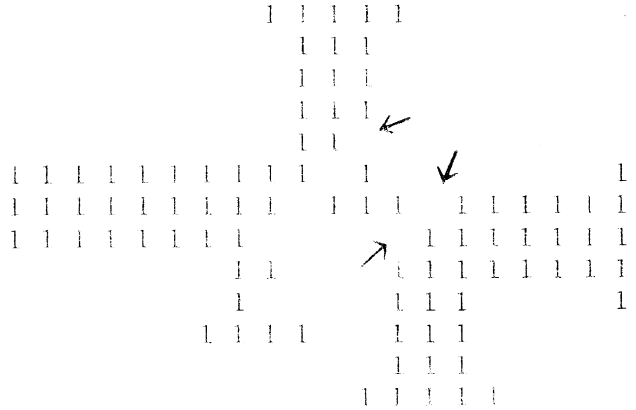
cycle 8



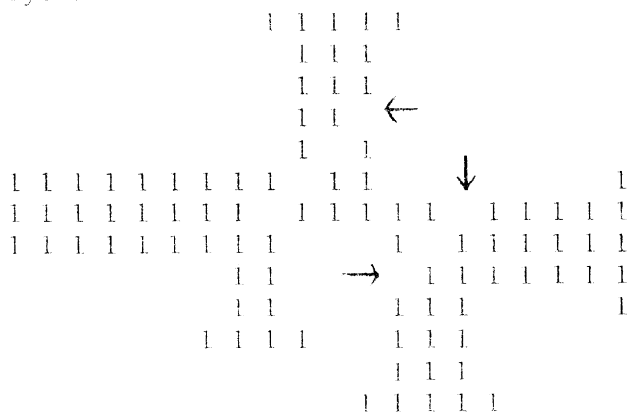
(continued)

In these three cycles, the signals have begun their exit from the fanout; however there is still a spurious signal in the short leg which must be annihilated later.

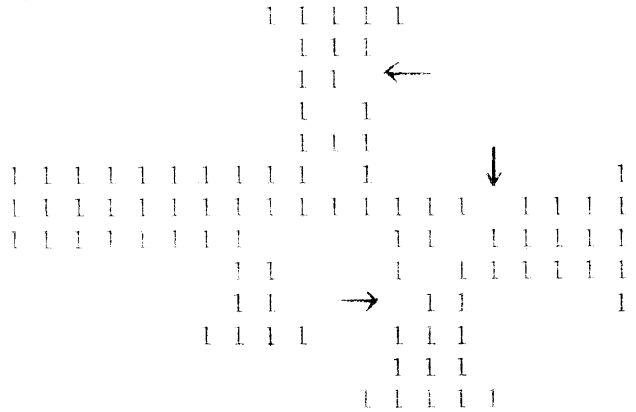
cycle 9



cycle 10



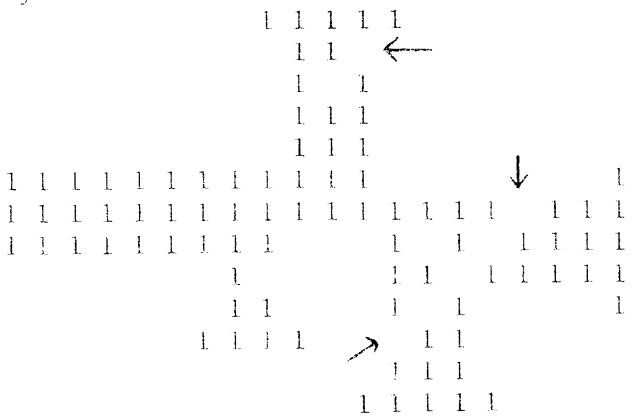
cycle 11



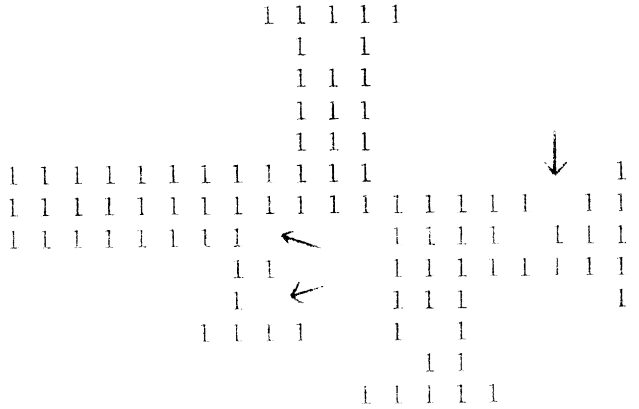
(continued)

In these three final cycles the exiting signals are well clear of the junction. In fact, the upward moving one encounters the dead-end and is completely destroyed. Note also the destruction of the signal in the short leg returning the fanout to its original state.

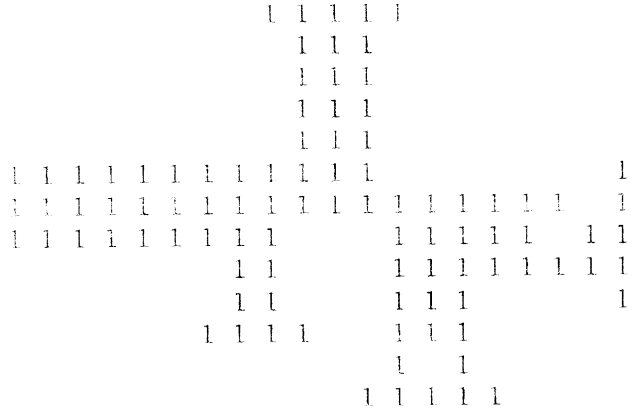
cycle 12



cycle 13



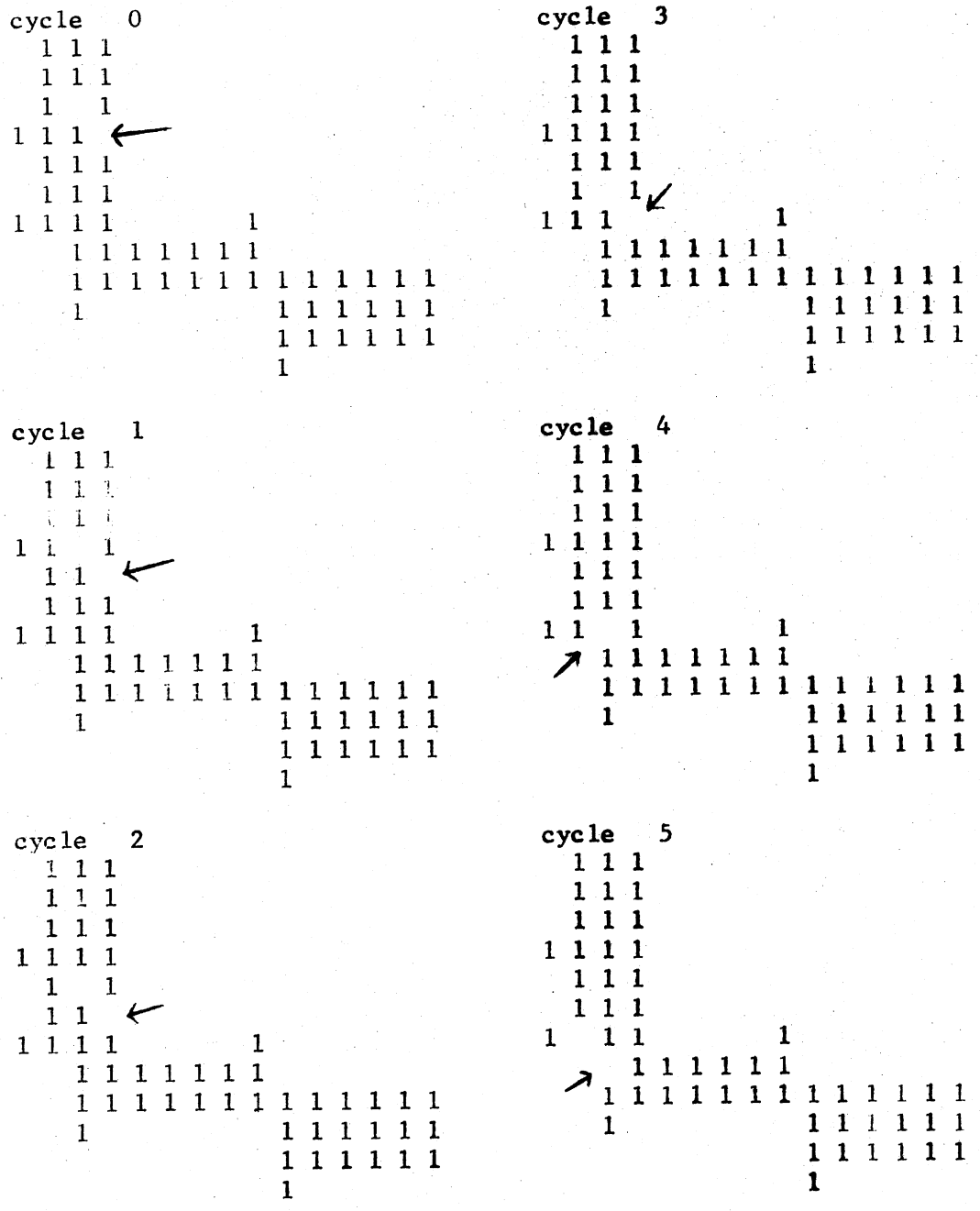
cycle 14



(END)

COMPUTER SIMULATION OF THE INSIDE-TO-INSIDE CURVE

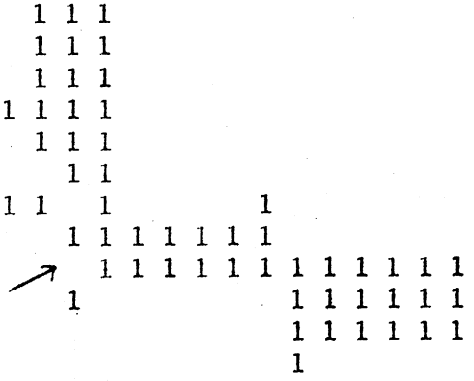
This simulation will show how the downward-moving signal will become a right-moving signal, both signals being on the inside side of the curve.



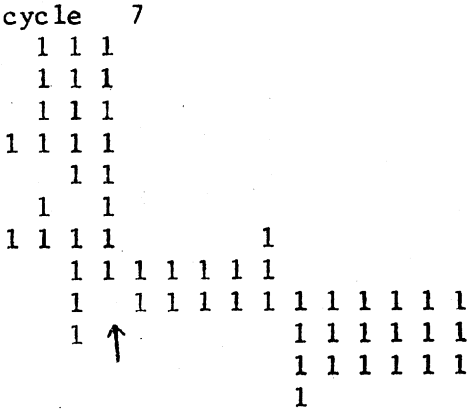
(continued)

Notice in cycles 6 and 7 that a spurious signal tries to return back up the input but it is halted by the extra 1. Notice also the peculiar type propagation in the section of width two. This type propagation was the basic type propagation for another version of two-state automaton.

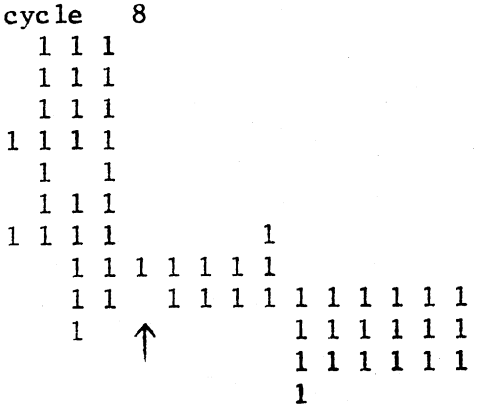
cycle 6



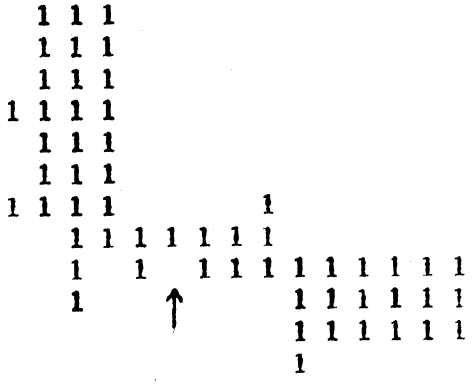
cycle 7



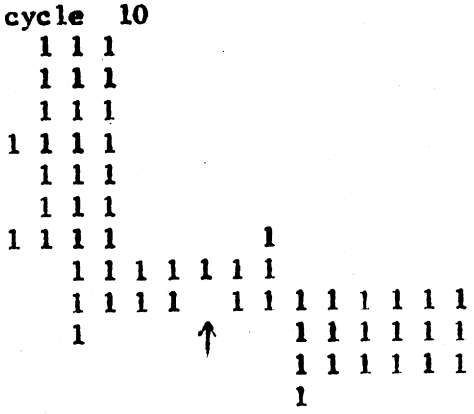
cycle 8



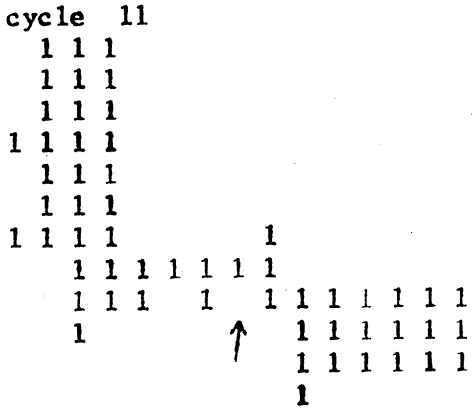
cycle 9



cycle 10



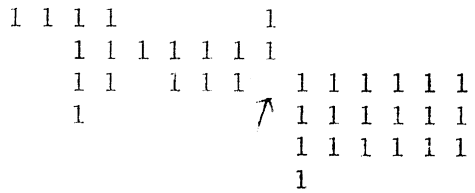
cycle 11



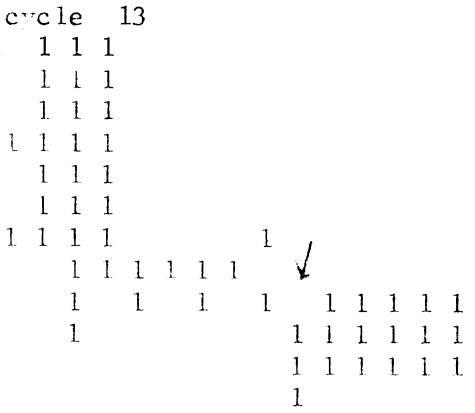
(continued)

In these final cycles the signal reaches the output wire.

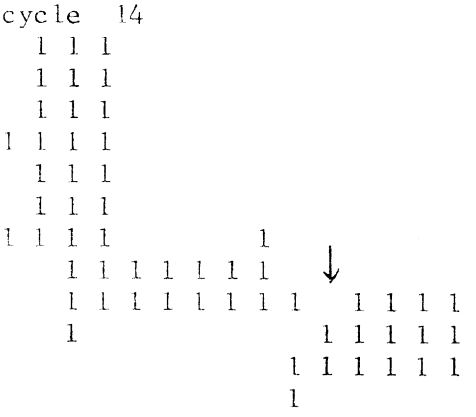
cycle 12



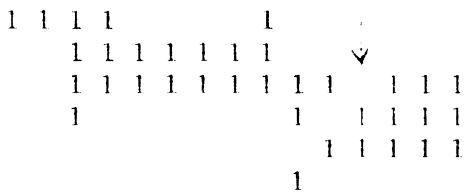
cycle 13



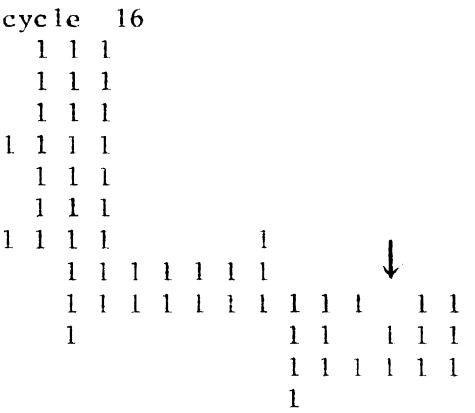
cycle 14



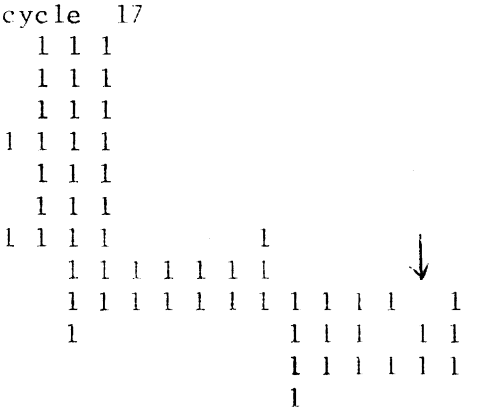
cycle 15



cycle 16



cycle 17



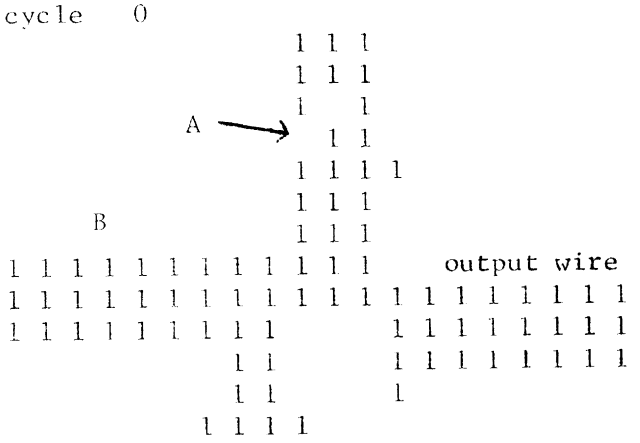
(END)

THE LOGIC FUNCTION: B AND NOT A

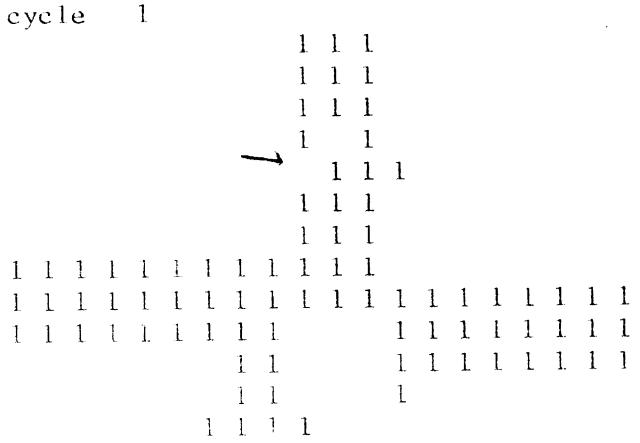
Case 1, A input only.

"B and not A" has the truth value false for A input only. Thus the signal moving downward should be destroyed and nothing should reach the exit wire on the right.

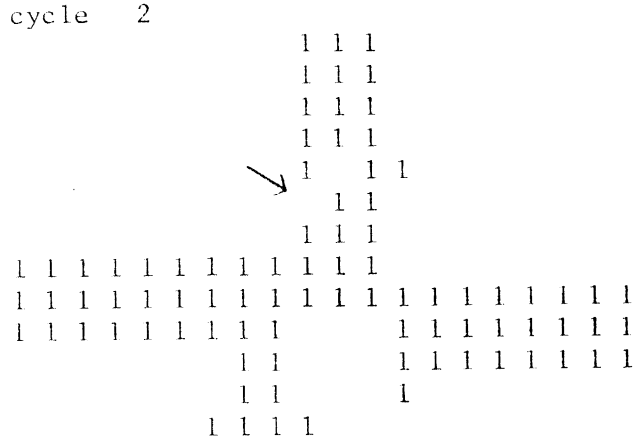
cycle 0



cycle 1



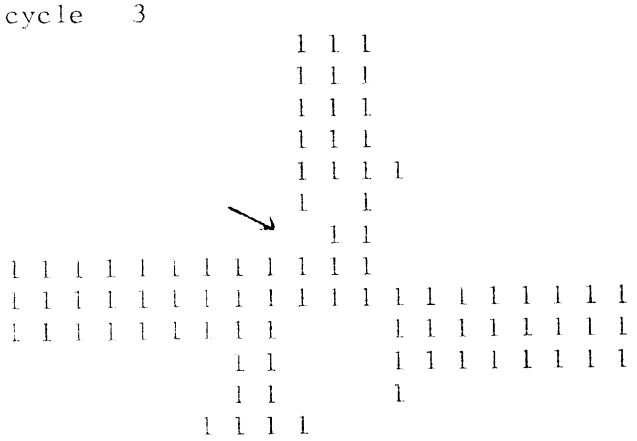
cycle 2



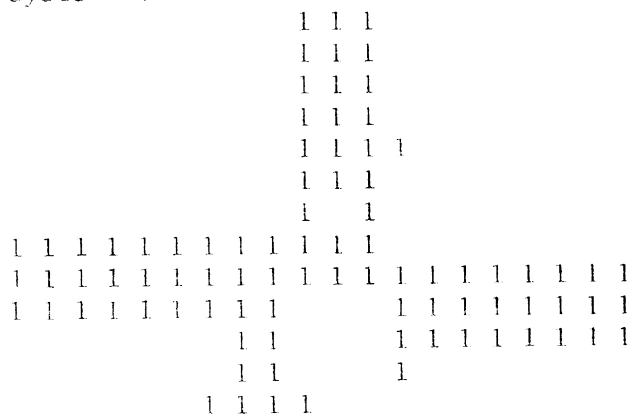
(continued)

The signal is truly destroyed in these three cycles and the logic element is left in its original position.

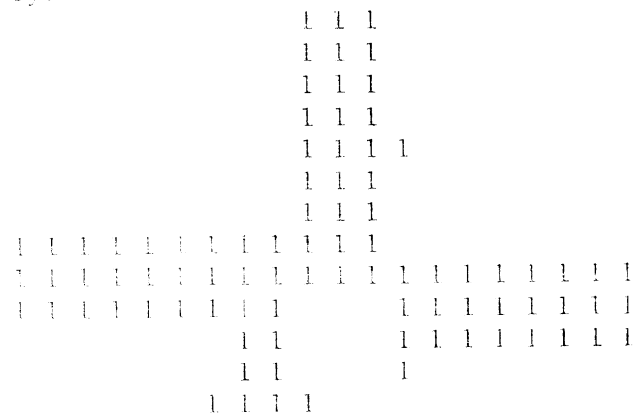
cycle 3



cycle 4



cycle 5



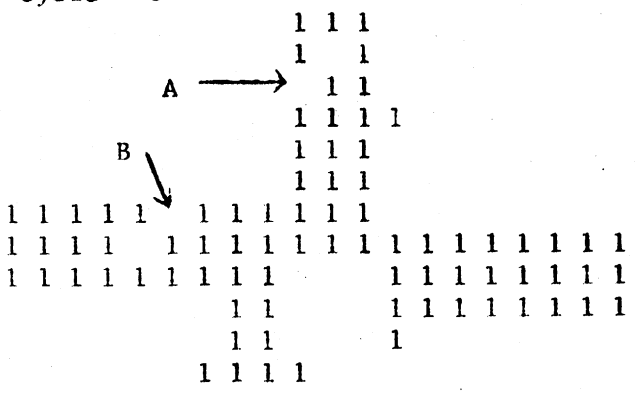
(END)

THE LOGIC FUNCTION B AND NOT A

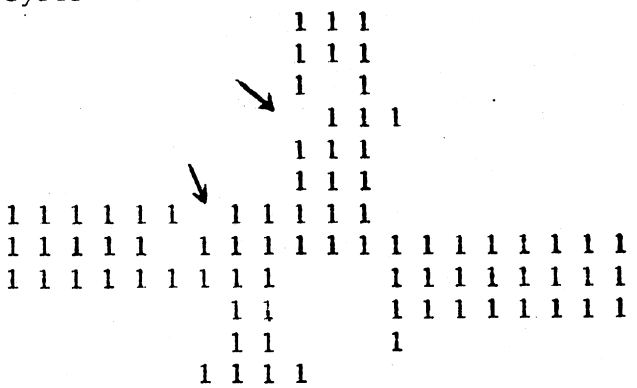
Case II, A and B Both Input.

Again the truth value of this function should be false or no output. These first three cycles show the signals entering the logic function element. Note the resemblence to the fanout element.

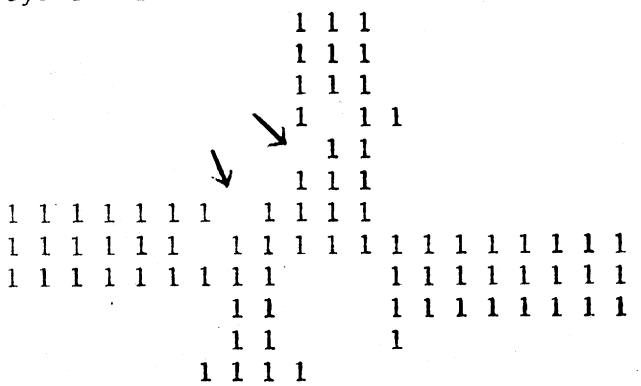
cycle 0



cycle 1



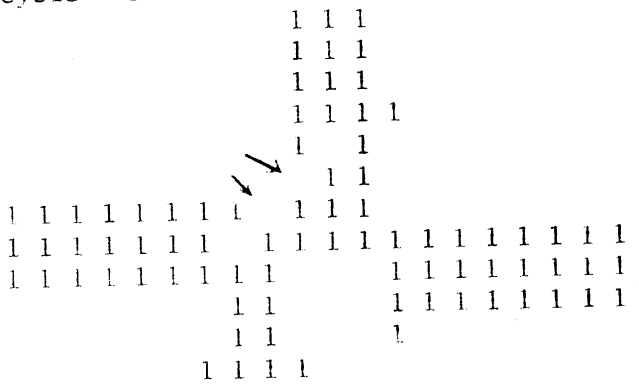
cycle 2



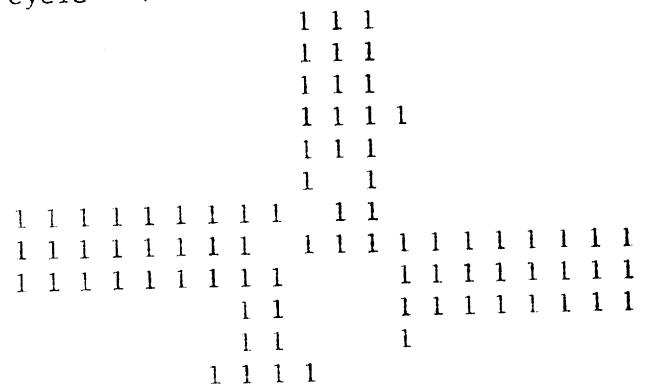
(continued)

These three cycles show the meeting of the two signals. Notice the introduction of the spurious signal into the short leg which must later disappear.

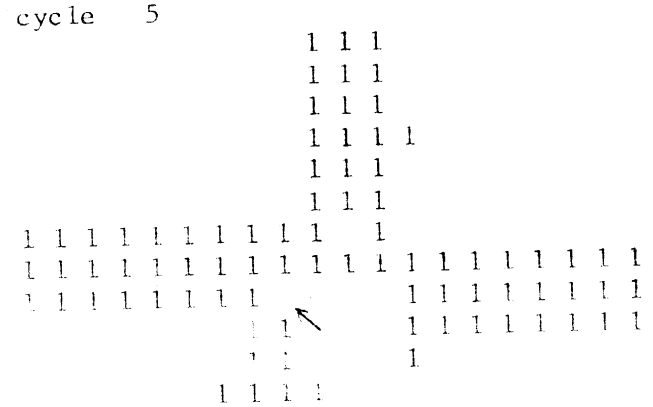
cycle 3



cycle 4



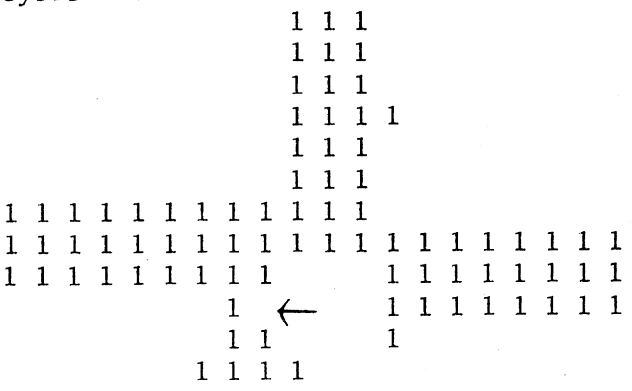
cycle 5



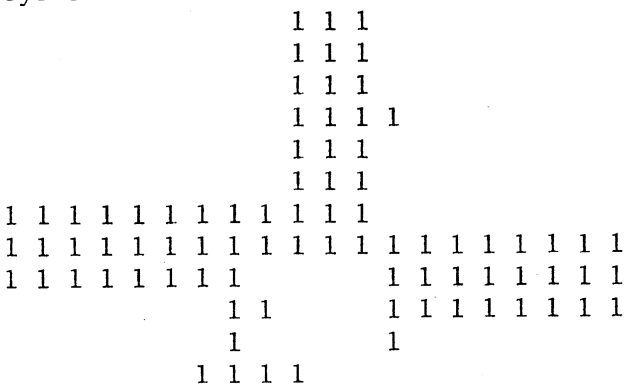
(continued)

These last three cycles show the anihilation of the spurious signal in the short leg. Thus no signal resulted in the output wire and the logic element returned to its original state.

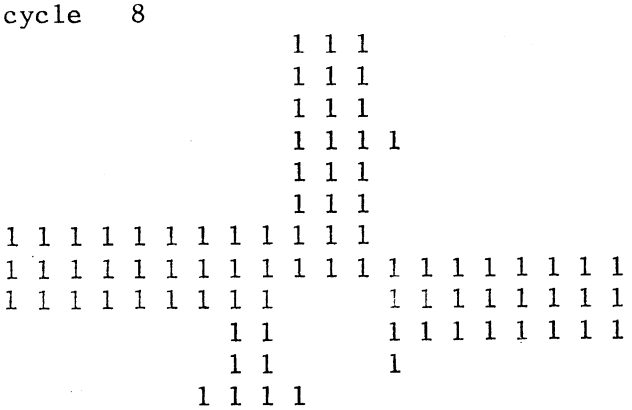
cycle 6



cycle 7



cycle 8



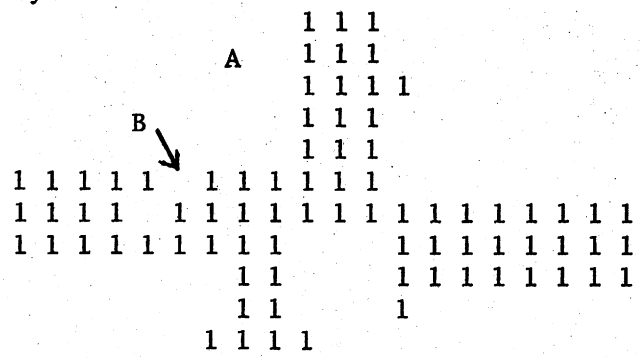
(END)

THE LOGIC FUNCTION B AND NOT A

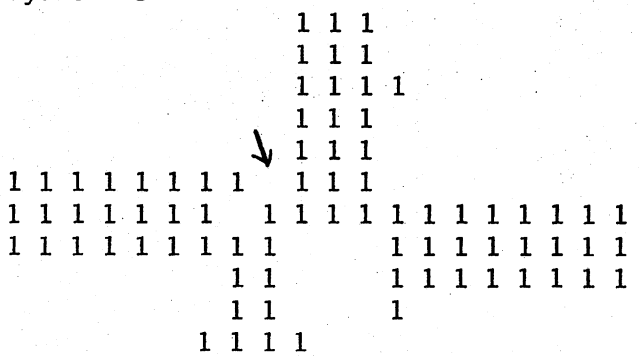
Case III, B Input Only.

This is the one case in which a signal should reach the output wire. The operation is exactly parallel to the fanout operation except that the signal trying to get out the A input is anihilated by the extra 1 on the right side of the A input wire. Only selected cycles are shown.

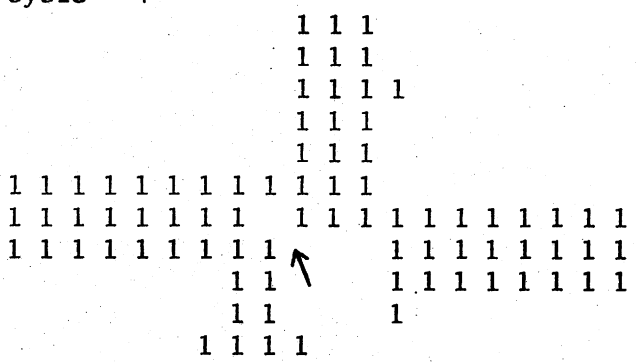
cycle 0



cycle 3



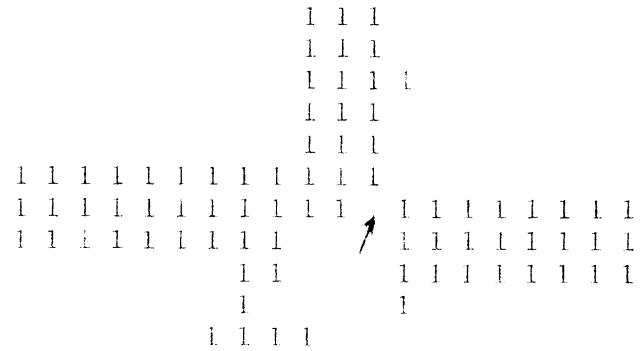
cycle 4



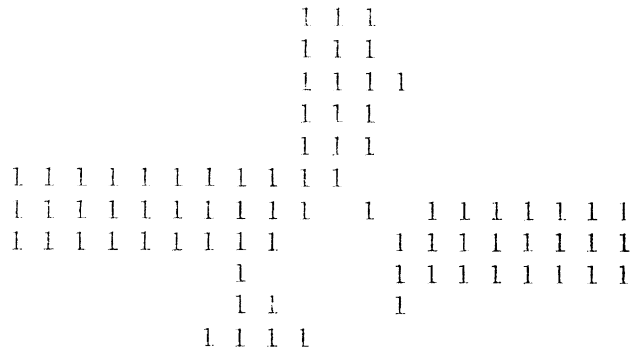
(continued)

Cycles 7, 8, and 9 show the signal reaching the output wire and the spurious signal starting up the A input wire.

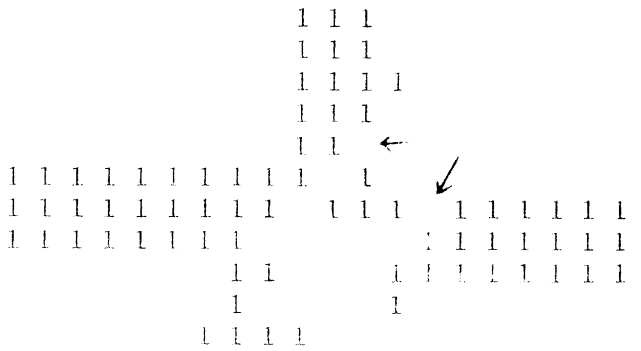
cycle 7



cycle 8



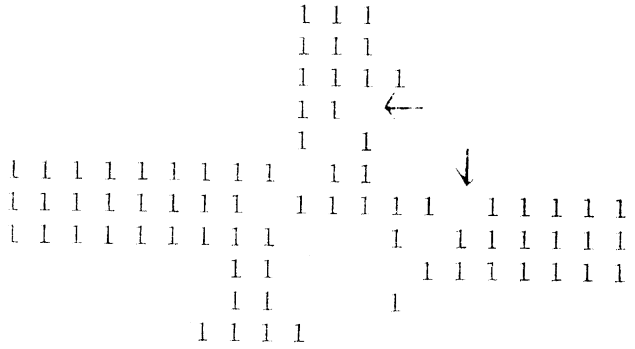
cycle 9



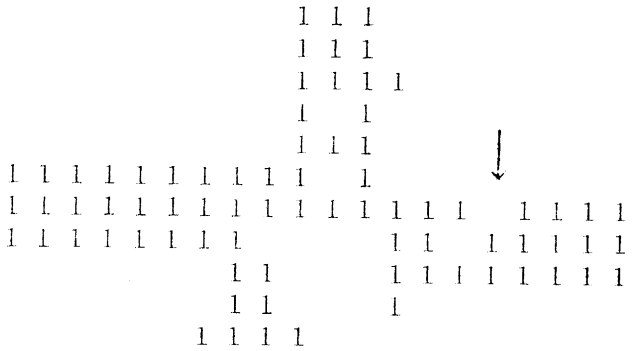
(continued)

Cycles 10, 11, and 12 show the destruction of the spurious signal and the output reaching the result wire. The spurious signal left in the short leg will anihilate itself two cycles later in cycle 14 (not shown).

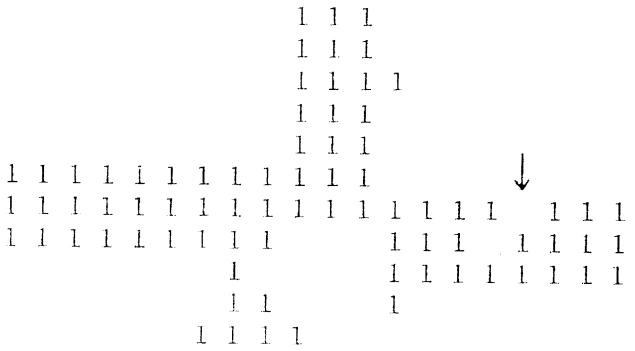
cycle 10



cycle 11



cycle 12

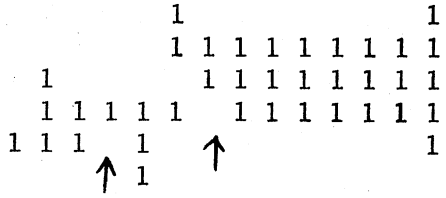


(END)

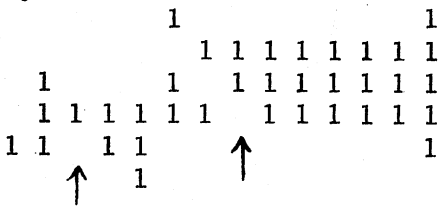
THE CLOCK

These cycles represent one full period of a clock. Clocks with other periods can also be constructed. This clock, however, has a period of eight cycles. The output is fed into a dead-end to destroy the signal, for the sake of this simulation only.

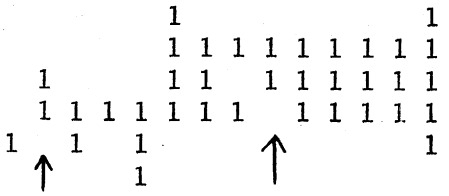
cycle 0



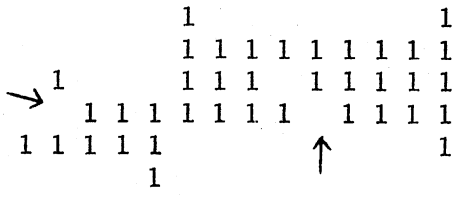
cycle 1



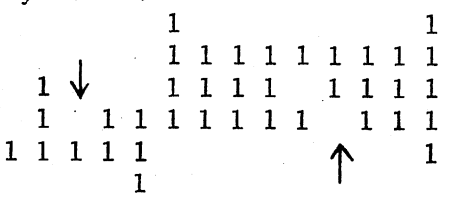
cycle 2



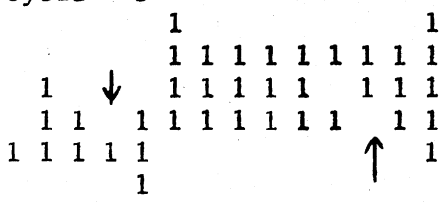
cycle 3



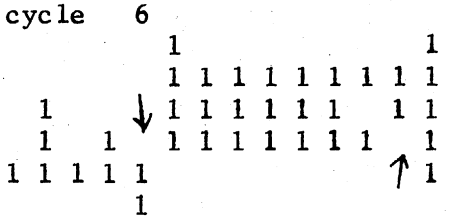
cycle 4



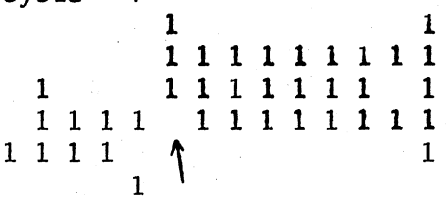
cycle 5



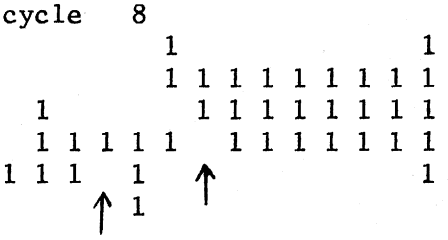
cycle 6



cycle 7



cycle 8



(END)

Notice the equivalence of cycles eight and zero.

APPENDIX II: THE FINITELY CONFIGURED THREE-STATE UNIVERSAL COMPUTER

The Transition Rules.

These rules are listed in the form "(CNESWR)" where C represents the current state of the cell, N the state of the North neighbor, East, South and West neighbors similarly abbreviated, and finally R represents the result state. Only transition rules are listed, i.e. C is different from R. Also, since the rules are isotropic, only one rule of each family is listed--the rotations and reflections possible with the neighbors N, E, S, and W are not all shown. (The quiescent state is denoted by blank.)

(1 2)
(11122)
(112 2)
(12 2)
(12 22)
(12222)
(222 1)
(22222)
(21 12)
(11 2)
(11222)
(1222)
(122)
(2 1)
(21112)
(211221)
(212221)

Computer Simulation of the Wire with Signal and the Dead-end.

Notice the disappearance of the signal in cycles 5 and 6.

```
2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2   1 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2

cycle   1

2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2   1 2 2 2 2
2 2 2 1 2 2 2 2 2 2 2 2

cycle   2

2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2   1 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2

cycle   3

2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2   1 2 2
2 2 2 2 2 2 2 2 2 2 2 2

cycle   4
2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2   1 2
2 2 2 2 2 2 2 2 2 2 2 2

cycle   5

2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2   2
2 2 2 2 2 2 2 2 2 2 2 2

cycle   6

2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2

(END)
```

Simulation of the Fanout or Junction.

The signal entering from the left will be seen to leave from the other three wires in cycles 9 and 10.

```
cycle   0
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2   1 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   1
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2   1 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2   1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   3
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2   1 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   4
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 1 2 2 2 2 2 2 2
2 2 2 2 2   1 2 2 2 2 2 2
2 2 2 2 2 1 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   5
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2   1 2 2 2 2 2 2
2 2 2 2 2 2   1 2 2 2 2 2
2 2 2 2 2   1 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

(continued)

In these remaining cycles, the signals exit the other three wires and enter dead-end wires where they are anihilated.

```
cycle   6
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 1 2
2 2 2 2 2 2   1 2 2 2 2 2
2 2 2 2 2 2 2   1 2 2 2 2
2 2 2 2 2 2   1 2 2 2 2 2
          2 1 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   7
          2 2 2
          2 2 2
          2 2 2
          2 1 2
          2   2
2 2 2 2 2 2     2 2 2 2 2
2 2 2 2 2 2 2 2   1 2 2 2
2 2 2 2 2 2     2 2 2 2 2
          2   2
          2 1 2
          2 2 2
          2 2 2
          2 2 2
```

```
cycle   8
          2 2 2
          2 2 2
          2 1 2
          2   2
          2 2 2
2 2 2 2 2 2   1 2 2 2 2 2
2 2 2 2 2 2 2 2 2   1 2 2
2 2 2 2 2 2   1 2 2 2 2 2
          2 2 2
          2   2
          2 1 2
          2 2 2
          2 2 2
```

```
cycle   9
            2 2 2
            2 1 2
            2   2
            2 2 2
            2 2 2
2 2 2 2 2 2 2   2 2 2 2 2
2 2 2 2 2 2 2 2 2 2   1 2
2 2 2 2 2 2 2   2 2 2 2 2
            2 2 2
            2 2 2
            2   2
            2 1 2
            2 2 2
```

```
cycle  10
            2 2 2
            2   2
            2 2 2
            2 2 2
            2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2   2
2 2 2 2 2 2 2 2 2 2 2 2 2
            2 2 2
            2 2 2
            2 2 2
            2   2
            2 2 2
```

```
cycle  11
            2 2 2
            2 2 2
            2 2 2
            2 2 2
            2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
            2 2 2
            2 2 2
            2 2 2
            2 2 2
            2 2 2
```

(END)

Simulation of the Two-Input/Two-Output Property of the Junction

It is seen in these simulations that two inputs at right angles will result in two output signals on the other two wires, here dead-ended.

```
cycle   0
        2 2 2
        2   2
        2 1 2
        2 2 2
2 2 2 2 2 2 2 2 2 2 2
2   1 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   1
        2 2 2
        2 2 2
        2   2
        2 1 2
2 2 2 2 2 2 2 2 2 2 2
2 2   1 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   2
        2 2 2
        2 2 2
        2 2 2
        2   2
2 2 2 2 2 1 2 2 2 2 2
2 2 2   1 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   3
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 1   1 2 2 2 2
2 2 2 2   1 2 2 2 2 2
2 2 2 2 1 2 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   4
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2   2   2 2 2 2
2 2 2 2 2   1 2 2 2 2
2 2 2 2   1 2 2 2 2 2
        2 2 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   5
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2     1 2 2 2
2 2 2 2 2   1 2 2 2 2
        2 1 2
        2 2 2
        2 2 2
        2 2 2
```

```
cycle   6
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2   2   1 2 2
2 2 2 2 2 2   2 2 2 2
        2   2
        2 1 2
        2 2 2
        2 2 2
```

```
cycle   7
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2   1 2
2 2 2 2 2 2 2 2 2 2 2
        2 2 2
        2   2
        2 1 2
        2 2 2
```

```
cycle   8
        2 2 2
        2 2 2
        2 2 2
        2 2 2
2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2   2
2 2 2 2 2 2 2 2 2 2 2
        2 2 2
        2 2 2
        2   2
        2 2 2
```

(END)

The Three-Input Anihilation Property of the Junction

In is seen in these computer simulations that three signals arriving simultaneously are mutually anihilated. Cycle 5 is seen to be the original junction configuration.

```
cycle   0
          2 2 2
          2 2 2
          2   2
          2 1 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2   1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 1 2
          2   2
          2 2 2
          2 2 2

cycle   1
          2 2 2
          2 2 2
          2 2 2
          2   2
          2 1 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2   1 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 1 2
          2   2
          2 2 2
          2 2 2
          2 2 2

cycle   2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2   2
2 2 2 2 2 2 1 2 2 2 2 2 2
2 2 2 2   1 2 2 2 2 2 2 2
2 2 2 2 2 2 1 2 2 2 2 2 2
          2   2
          2 2 2
          2 2 2
          2 2 2
          2 2 2

cycle   3
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 1   1 2 2 2 2 2 2
2 2 2 2 2     2 2 2 2 2 2 2
2 2 2 2 2 1   1 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2

cycle   4
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2   2   2 2 2 2 2 2
2 2 2 2 2 2   2 2 2 2 2 2 2
2 2 2 2 2   2   2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2

cycle   5
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
          2 2 2
```

(END)

Simulation of the Extensible Wire.

The signal seen propagating to the right in cycle 0 will reflect from the end of this wire (due to the extra "2" distinguishing this wire from the dead-end configuration) generating an echo signal as seen in cycles 8 and 9. The wire is also lengthened by two cells.

```
cycle   0

2 2 2 2 2 2 2
2 2 2   1 2 2 2
2 2 2 2 2 2 2

cycle   1

2 2 2 2 2 2 2
2 2 2 2   1 2 2
2 2 2 2 2 2 2

cycle   2

2 2 2 2 2 2 2
2 2 2 2 2   1 2
2 2 2 2 2 2 2

cycle   3

2 2 2 2 2 2 2
2 2 2 2 2 2   2
2 2 2 2 2 2 2

cycle   4

2 2 2 2 2 2 2
2 2 2 2 2 2 2 1
2 2 2 2 2 2 2

cycle   5

2 2 2 2 2 2 2 2
2 2 2 2 2 2 1 1 2
2 2 2 2 2 2 2 2

cycle   6

2 2 2 2 2 2 2 2
2 2 2 2 2 1     2
2 2 2 2 2 2 2 2

cycle   7

2 2 2 2 2 2 2 2
2 2 2 2 1   2 1 1
2 2 2 2 2 2 2 2

cycle   8

2 2 2 2 2 2 2 2 2
2 2 2 1   2 2   2 2
2 2 2 2 2 2 2 2 2

cycle   9

2 2 2 2 2 2 2 2 2
2 2 1   2 2 2 2 2 2
2 2 2 2 2 2 2 2 2

cycle  10

2 2 2 2 2 2 2 2 2
2 1   2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2

(END)
```

Simulation of the Meeting of Two Signals

Two signals seen meeting in phase with each other will mutually anihilate each other.

```
cycle   0

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2   1 2 2 2 1   2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2


cycle   1

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2   1 2 1   2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2


cycle   2

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2   2   2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2


cycle   3

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2


(END)
```

APPENDIX III: THE TWO-STATE, NINE-NEIGHBOR FINITELY CONFIGURED UNIVERSAL COMPUTER

The Transition Rules.

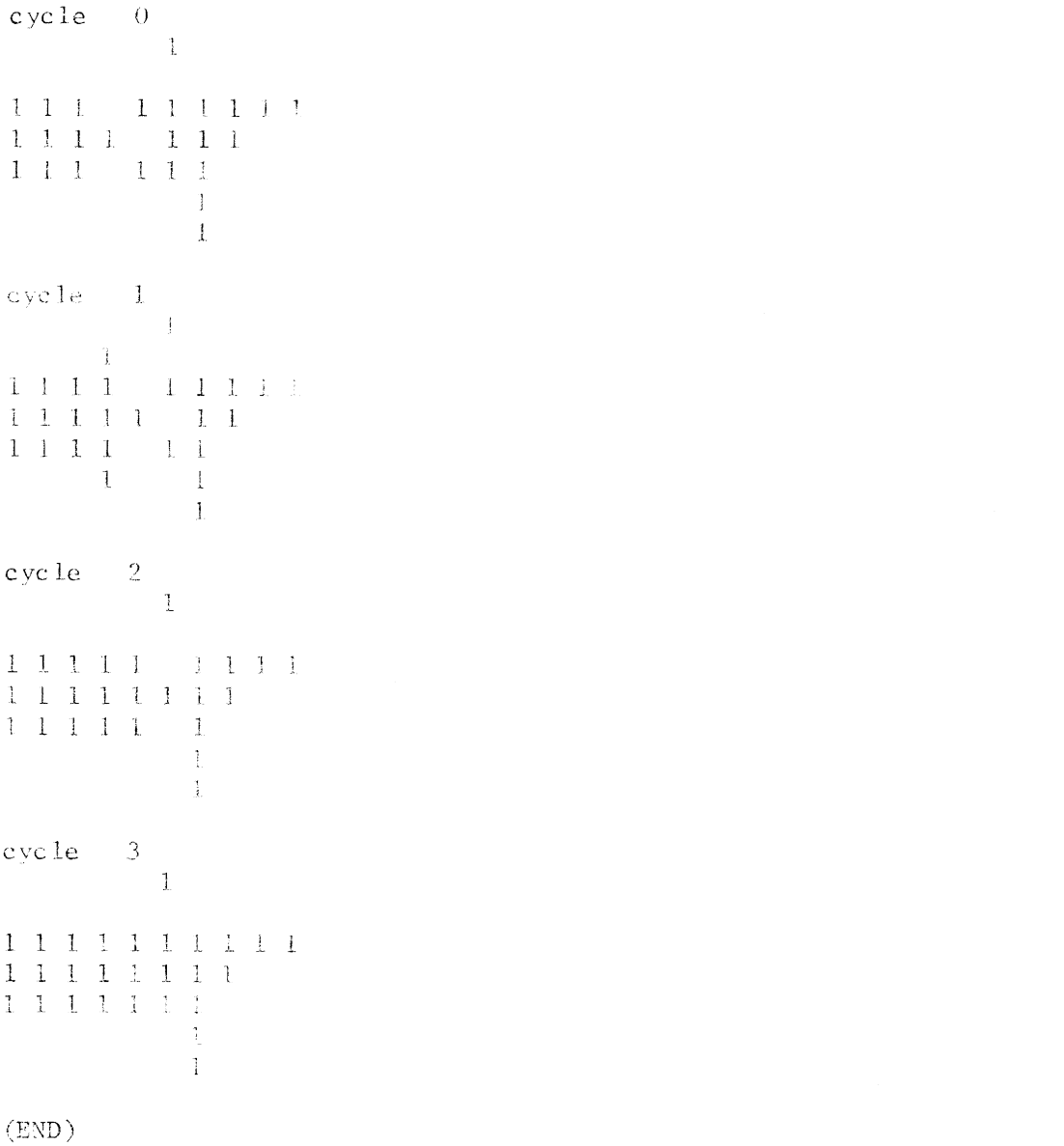
These rules are listed in the form "(CabcdefghR)" where the letters "abcdefgh" represent the neighborhood taken either clockwise or counter-clockwise from any corner. See Appendix II for further explanation.

(11111111)	p*
(11 11)	p
(1111 1 11)	p
(11 1 11)	p
(1 1 1)	e
(1111 1 1)	e
(111111 1)	e
(1 1 1 1 1)	e
(11 11)	e
(11 1)	e
(11 1 1 11)	e
(111)	e
(11111 1)	e
(1111 1)	e
(1 1 1)	e
(1 1 1 1)	e
(11 11 1)	e
(11 1 1 1)	e
(111111111)	j
(11111111)	j
(11 1 11)	j
(11 11 1)	j
(11 1 1 11)	j
(1111 11)	j
(111111 1 1)	j
(11 1 1 11)	j
(111 11)	j
(111 11 1)	j
(11 1 1)	j
(111 11 11)	j
(1111 11 1)	j
(111 111 1)	j

* The letter after each transition rule indicates the purpose for which the rule was created. P corresponds to signal propagation, e to echo and j for the junction properties.

Simulation of the Signal Entering the Dead-end.

The right-moving signal of cycle zero is seen to be anihilated in cycle 3 of this dead-end wire.



Simulation of the Extensible Wire.

The right-moving signal reaches the end of the extensible wire with the results of extending the wire and of generating an echo signal. Since the signal can reach the end of this wire in two different phases, both cases are shown.

```
cycle   0

      1
1 1 1 1   1 1
1 1 1 1 1   1 1
1 1 1 1   1 1
      1

cycle   1

1 1 1 1 1   1
1 1 1 1 1 1 1 1
1 1 1 1 1   1

cycle   2

          1
1 1 1 1 1 1 1
1 1 1 1 1 1   1
1 1 1 1 1 1 1
          1

cycle   3

          1
1 1 1 1 1 1   1
1 1 1 1 1   1
1 1 1 1 1 1   1
          1

cycle   4

1 1 1 1 1   1 1
1 1 1 1   1 1 1 1
1 1 1 1 1   1 1

cycle   5

          1
1 1 1 1   1 1 1
1 1 1   1 1 1 1 1
1 1 1 1   1 1 1
          1

(END)
```

```
cycle   0

1 1 1 1   1 1
1 1 1 1 1   1 1
1 1 1 1   1 1

cycle   1

          1
1 1 1 1 1   1
1 1 1 1 1 1 1 1
1 1 1 1 1   1
          1

cycle   2

1 1 1 1 1 1 1
1 1 1 1 1 1   1
1 1 1 1 1 1 1

cycle   3

1 1 1 1 1 1   1
1 1 1 1 1   1
1 1 1 1 1 1   1

cycle   4

            1
1 1 1 1 1   1 1
1 1 1 1   1 1 1 1
1 1 1 1 1   1 1
            1

cycle   5

1 1 1 1   1 1 1
1 1 1   1 1 1 1 1
1 1 1 1   1 1 1

(END)
```

Simulation of the Fanout Property of the Junction.

The signal seen entering the junction from the top wire is seen to exit the other three wires. For brevity, the signal is shown almost into the junction in cycle 0.

```
cycle   0
          1 1 1
            1
      1     1     1     1
          1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
          1 1 1
      1   1 1 1     1
          1 1 1
          1 1 1
```

```
cycle   1
          1 1 1
          1 1 1
      1     1       1
          1     1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
          1 1 1
      1   1 1 1     1
          1 1 1
          1 1 1
```

```
cycle   2
          1 1 1
          1 1 1
      1   1 1 1     1
            1
1 1 1 1 1     1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
          1 1 1
      1   1 1 1     1
          1 1 1
          1 1 1
```

```
cycle   3
            1 1 1
            1 1 1
      1     1 1 1     1
              1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1     1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
            1 1 1
      1     1 1 1     1
            1 1 1
            1 1 1
```

```
cycle   4
            1 1 1
            1 1 1
      1     1 1 1     1
            1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1       1       1 1 1 1
1 1 1 1 1     1 1 1 1 1
            1 1 1
      1     1 1 1     1
            1 1 1
            1 1 1
```

```
cycle   5
            1 1 1
            1 1 1
      1     1 1 1     1
            1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1       1 1 1       1 1 1
1 1 1 1       1       1 1 1 1
            1 1 1
      1     1 1 1     1
            1 1 1
            1 1 1
```

```
cycle   6
              1 1 1
              1 1 1
      1       1 1 1       1
              1 1 1
1 1 1         1 1 1         1 1 1
1 1     1 1 1 1 1 1       1 1
1 1 1         1 1 1         1 1 1
                1
      1       1     1       1
              1 1 1
              1 1 1
```

```
cycle   7
              1 1 1
              1 1 1
      1       1 1 1       1
              1 1 1
1 1     1 1 1 1 1 1       1 1
1     1 1 1 1 1 1 1 1     1
1 1     1 1 1 1 1 1       1 1
              1 1 1
      1         1         1
              1     1
              1 1 1
```

```
cycle   8
              1 1 1
              1 1 1
      1       1 1 1       1
              1 1 1
1     1 1 1 1 1 1 1 1     1
  1 1 1 1 1 1 1 1 1 1
1     1 1 1 1 1 1 1 1     1
              1 1 1
      1       1 1 1       1
                1
              1     1
```

(END)

Simulation of the Two-Input Anihilation Property of the Junction.

Two signals seen entering the junction in cycle 0 are mutually anihilated in cycles 3 and 4.

```
cycle   0
          1 1 1
          1 1 1
            1
      1     1     1     1
          1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
          1 1 1
      1     1     1     1
            1
          1 1 1
          1 1 1
```

```
cycle   1
          1 1 1
          1 1 1
          1 1 1
      1       1       1
          1     1
1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1
          1     1
      1       1       1
          1 1 1
          1 1 1
          1 1 1
```

```
cycle   2
          1 1 1
          1 1 1
          1 1 1
      1     1 1 1     1
            1
1 1 1 1 1 1     1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1     1 1 1 1 1 1
            1
      1     1 1 1     1
          1 1 1
          1 1 1
          1 1 1
```

```
cycle   3
          1 1 1
          1 1 1
          1 1 1
      1     1 1 1     1
            1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1
      1     1 1 1     1
          1 1 1
          1 1 1
          1 1 1
```

```
cycle   4
          1 1 1
          1 1 1
          1 1 1
      1     1 1 1     1
          1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
          1 1 1
      1     1 1 1     1
          1 1 1
          1 1 1
          1 1 1
```

(END)

Simulation of the Three-Input Property of the Junction.

Three simultaneous inputs entering the junction are anihilated as seen in cycles 3 to 4.

```
cycle   0
          1 1 1
          1 1 1
            1
      1     1     1     1
            1 1 1
1 1     1 1 1 1 1 1 1 1 1
1 1 1     1 1 1 1 1 1 1 1
1 1     1 1 1 1 1 1 1 1 1
            1 1 1
      1     1     1     1
            1
            1 1 1
            1 1 1
```

```
cycle   1
          1 1 1
          1 1 1
          1 1 1
      1       1       1
            1   1
1 1 1     1 1 1 1 1     1 1 1
1 1 1 1     1 1 1     1 1 1 1
1 1 1     1 1 1 1 1     1 1 1
            1   1
      1       1       1
          1 1 1
          1 1 1
          1 1 1
```

```
cycle   2
          1 1 1
          1 1 1
          1 1 1
      1   1 1 1     1
            1
1 1 1 1     1     1 1 1 1 1 1
1 1 1 1 1     1 1 1 1 1 1 1
1 1 1 1     1     1 1 1 1 1 1
            1
      1     1 1 1     1
          1 1 1
          1 1 1
          1 1 1
```

```
cycle   3
            1 1 1
            1 1 1
            1 1 1
        1   1 1 1     1
            1 1
1 1 1 1 1       1 1 1 1 1 1 1
1 1 1 1 1       1 1 1 1 1 1 1
1 1 1 1 1       1 1 1 1 1 1 1
            1 1
        1   1 1 1     1
            1 1 1
            1 1 1
            1 1 1
```

```
cycle   4
            1 1 1
            1 1 1
            1 1 1
        1   1 1 1     1
            1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 1 1
        1   1 1 1     1
            1 1 1
            1 1 1
            1 1 1
```

(END)

Simulation of the Operation of the Phase Convertor Element.

The phase convertor's purpose is to make the phase of a signal uniform regardless of its phase on entry. Here we illustrate both cases of entry phase with one signal by putting the upper half of the signal in one phase (an extra 1) and the lower half in the other.

```
cycle   0
              1
  1
1 1   1 1 1 1 1 1 1 1 1
1 1 1   1 1 1 1 1 1 1 1
1 1   1 1 1 1 1 1 1 1 1

              1

cycle   1
              1

1 1 1   1 1 1 1 1 1 1 1
1 1 1 1   1 1 1 1 1 1 1
1 1 1   1 1 1 1 1 1 1 1
    1
              1

cycle   2
              1
      1
1 1 1 1   1 1 1 1 1 1 1
1 1 1 1 1   1 1 1 1 1 1
1 1 1 1   1 1 1 1 1 1 1

              1

cycle   3
              1

1 1 1 1 1   1 1 1 1 1 1
1 1 1 1 1 1   1 1 1 1 1
1 1 1 1 1   1 1 1 1 1 1
        1
              1

cycle   4
              1

1 1 1 1 1 1   1 1 1 1 1
1 1 1 1 1 1 1   1 1 1 1
1 1 1 1 1 1   1 1 1 1 1

              1

cycle   5
                  1

1 1 1 1 1 1 1   1 1 1 1
1 1 1 1 1 1 1 1   1 1 1
1 1 1 1 1 1 1   1 1 1 1

                  1

cycle   6
                  1

1 1 1 1 1 1 1 1   1 1 1
1 1 1 1 1 1 1 1 1   1 1
1 1 1 1 1 1 1 1   1 1 1

                  1

cycle   7
                  1
                    1
1 1 1 1 1 1 1 1 1   1 1
1 1 1 1 1 1 1 1 1 1   1
1 1 1 1 1 1 1 1 1   1 1
                    1
                  1

cycle   8
                  1

1 1 1 1 1 1 1 1 1 1   1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1

                  1

(END)
```

APPENDIX IV: THE FOUR-STATE UNIVERSAL CONSTRUCTOR

The Transition Rules.

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

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

Computer Simulation of the Signal in the Wire and the Dead-end.

The signal seen moving to the right is destroyed in cycles 4 and 5.

```
cycle   0

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

cycle   1

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

cycle   2

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

cycle   3

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

cycle   4

x x x x x x x x x
                2 x
x x x x x x x x x

cycle   5

x x x x x x x x x
                  x
x x x x x x x x x
```

The Simulation of the Fanout Property of the Junction.

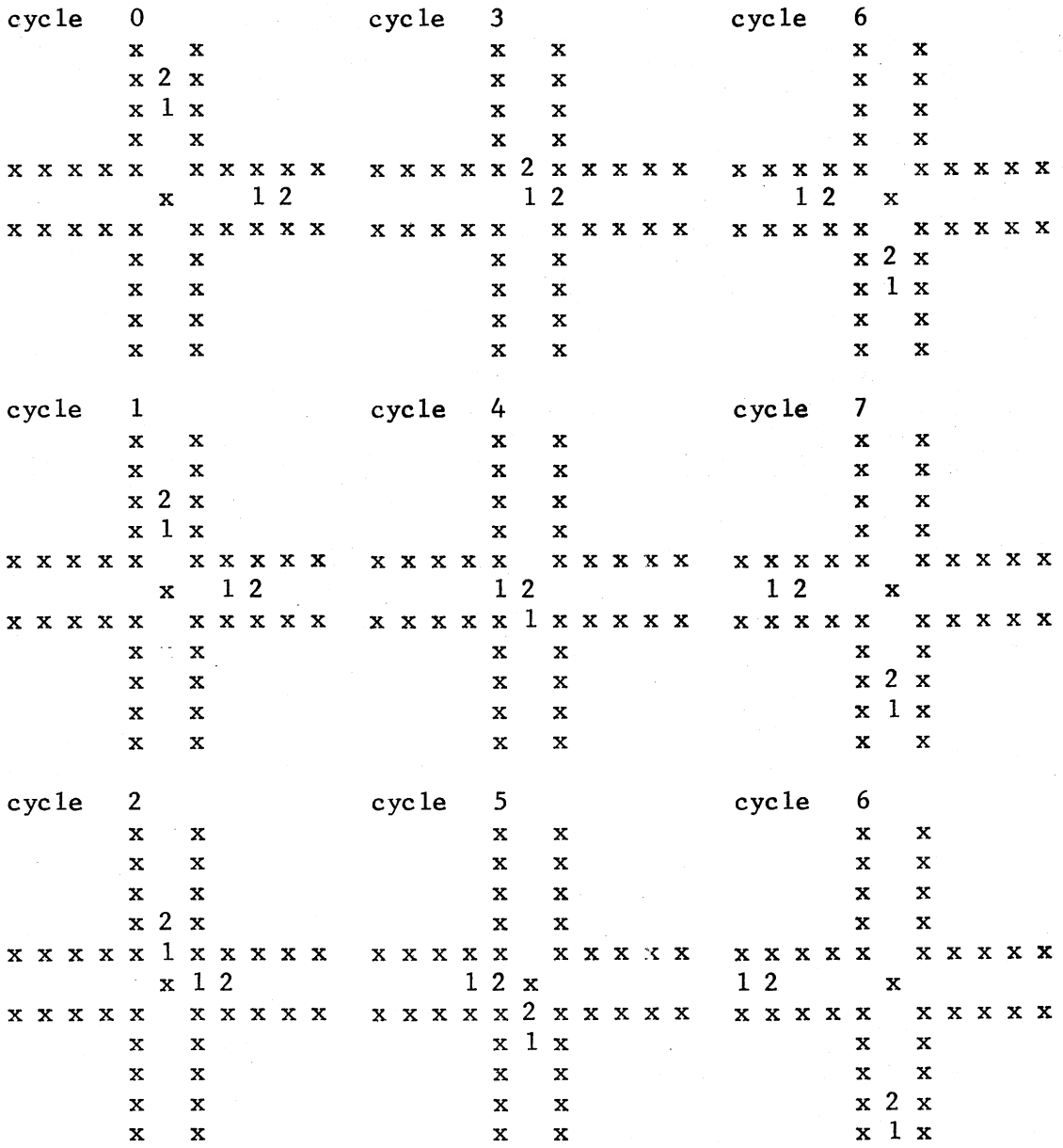
The signal seen entering from the left in cycle 0 fans out the other three wires in cycles 7 and 8. Notice the delay of one cycle in cycles 3 and 4.

```
cycle  0                cycle  3                cycle  6
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x
       x   x                   x   x                   x 1 x
x x x x x   x x x x x   x x x x x   x x x x x   x x x x x 2 x x x x x
  2 1     x                       2 2                     x 2 1
x x x x x   x x x x x   x x x x x   x x x x x   x x x x x 2 x x x x x
       x   x                   x   x                   x 1 x
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x

cycle  1                cycle  4                cycle  7
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x
       x   x                   x   x                   x 1 x
       x   x                   x   x                   x 2 x
x x x x x   x x x x x   x x x x x   x x x x x   x x x x x   x x x x x
    2 1   x                       2 1                     x   2 1
x x x x x   x x x x x   x x x x x   x x x x x   x x x x x   x x x x x
       x   x                   x   x                   x 2 x
       x   x                   x   x                   x 1 x
       x   x                   x   x                   x   x
       x   x                   x   x                   x   x

cycle  2                cycle  5                cycle  8
       x   x                   x   x                   x   x
       x   x                   x   x                   x 1 x
       x   x                   x   x                   x 2 x
       x   x                   x   x                   x   x
x x x x x   x x x x x   x x x x x 1 x x x x x   x x x x x   x x x x x
      2 1 x                       2 1                     x     2 1
x x x x x   x x x x x   x x x x x 1 x x x x x   x x x x x   x x x x x
       x   x                   x   x                   x   x
       x   x                   x   x                   x 2 x
       x   x                   x   x                   x 1 x
       x   x                   x   x                   x   x
```

(END)

Simulation of the Two-Input/Two-Output Property of the Junction.

The two signals entering the junction at right angles are seen exiting the other two wires in cycles 6 and 7. In this case there is no delay.



(END)

Simulation of the Anihilation Properties of the Junction

These two simulations illustrate that two signals entering the junction, not at right angles, and that three entering signals will be anihilated. The simulations begin with the signals almost in the junction.

```
cycle   0
        x   x
        x   x
        x   x
        x 2 x
x x x x x 1 x x x x x
      2 1 x 1 2
x x x x x   x x x x x
        x   x
        x   x
        x   x
        x   x

cycle   1
        x   x
        x   x
        x   x
        x   x
x x x x x 2 x x x x x
        2 x 2
x x x x x   x x x x x
        x   x
        x   x
        x   x
        x   x

cycle   2
        x   x
        x   x
        x   x
        x   x
x x x x x   x x x x x
          x
x x x x x   x x x x x
        x   x
        x   x
        x   x
        x   x

(END)
```

```
cycle   0
        x   x
        x   x
        x   x
        x 2 x
x x x x x 1 x x x x x
          x
x x x x x 1 x x x x x
        x 2 x
        x   x
        x   x
        x   x

cycle   1
        x   x
        x   x
        x   x
        x   x
x x x x x 2 x x x x x
          x
x x x x x 2 x x x x x
        x   x
        x   x
        x   x
        x   x

cycle   2
        x   x
        x   x
        x   x
        x   x
x x x x x   x x x x x
          x
x x x x x   x x x x x
        x   x
        x   x
        x   x
        x   x

(END)
```

Simulation of the Creation of Single Wing Signal from Logic Signal.

The "12" type logic signal entering this device in cycle 0 is seen to become a single wing signal propagating out the skinny wire in cycles 5 through 9.

```
cycle   0
      x
x x x x x
  2 1     x x x x x x x x
x x x x x
        x

cycle   1
      x
x x x x x
    2 1   x x x x x x x x
x x x x x
        x

cycle   2
      x
x x x x x
      2 1 x x x x x x x x
x x x x x
        x

cycle   3
      x
x x x x 1
        2 1 x x x x x x x
x x x x x
        x

cycle   4
      x
x x x x x 1
        2 x 1 x x x x x x
x x x x x
        x
```

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

cycle   6
      x
x x x x x     1
          x x x 1 x x x x
x x x x x
        x

cycle   7
      x
x x x x x       1
          x x x x 1 x x x
x x x x x
        x

cycle   8
      x
x x x x x
          x x x x x 1 x x
x x x x x
        x

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

(END)

Simulation of the Creation of the Constructing Arm.

The collision of two of the previous single wing signals creates the arm bud shown in cycle 8. Two more signals are seen to fill in the gap at the arm's base.

```
cycle   0

  1           1
x x 1 x x x 1 x x

cycle   1

    1       1
x x x 1 x 1 x x x

cycle   2

      1   1
x x x x x x x x x

cycle   3

        2
x x x x x x x x x

cycle   4

        x
      x   x
x x x x x x x x x

cycle   5

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

cycle   6

        2
      1 x 1
x x x x x x x x x
```

```
cycle   7
        x

        1
x x x x x x x x x

cycle   8
        x
        x

1 x x x x x x x 1

cycle   9
        x
        x
1               1
x 1 x x x x x 1 x

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

cycle  11
        x
        x
    1       1
x x x 1 x 1 x x x

cycle  12
        x
        x
      1   1
x x x x x x x x x

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

(END)

Simulation of the Lengthening of the Arm.

Two single-wing signals colliding at the arm's base cause a double-wing signal to propagate out the arm. This double-wing lengthens the arm and generates an echo signal.

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

cycle   1
x
x
x 1
x x x x x x
x 1
x
x

cycle   2
x
x
x
x 2 x x x x
x
x
x

cycle   3
x
x
x x
x 1 x x x x
x x
x
x

cycle   4
x
x
x 1
x x 1 x x x
x 1
x
x

cycle   5
x
x
x   1
x x x 1 x x
x   1
x
x

cycle   6
x
x
x     1
x x x x 1 x
x     1
x
x

cycle   7
x
x
x       1
x x x x x 2
x       1
x
x

cycle   8
x
x
x
x x x x 1   x
x
x
x

cycle   9
x
x
x
x x x 1   x x
x
x
x

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

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

(END)

The Process of Construction by Use of the Erasing-Wing Signal.

Normally the return of the echo signal as in the previous simulation results in a gap left at the base of the arm. This gap is filled as shown in an earlier simulation. However, if two more single-wing signals are synchronized with the return of the echo as shown in these simulations, then the erasing-wing is created. When this erasing wing reaches the arm's end, it deposits an x just beyond the end of where the arm was.

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

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

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

```
cycle   3
x
x
x x
x 1 1   x x x x x
x x
x
x
```

```
cycle   4
x
x
x 1
x   1 x x x x x x
x 1
x
x
```

```
cycle   5
x
x
x   1
x 2   1 x x x x x
x   1
x
x
```

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

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

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

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

```
cycle  10
x
x
x             1
x               2
x             1
x
x
```

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

(END)

Simulation of Three Cases of the Erasing-Wing Reaching the Arm's End.

Three cases are shown for the erasing-wing signal reaching the end of the arm. These three cases correspond to the allowable configurations of x's already in the construction zone. The case of the arm's having to deposit an x between two existing x's never occurs.

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

cycle   1
  1
    1 x x x   x
  1

cycle   2
    1
      1 x x   x
    1

cycle   3
      1
        1 x   x
      1

cycle   4
        1
          2   x
        1

cycle   5
              x x


(END)
```

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

cycle   1
  1         x
    1 x x x
  1

cycle   2
    1       x
      1 x x
    1

cycle   3
      1     x
        1 x
      1

cycle   4
        1   x
          2
        1

cycle   5
                x
                x


(END)
```

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

cycle   1
  1         x
    1 x x x   x
  1

cycle   2
    1       x
      1 x x   x
    1

cycle   3
      1     x
        1 x   x
      1

cycle   4
        1   x
          2   x
        1

cycle   5
                x
                x x


(END)
```

Simulation of the Activation Element.

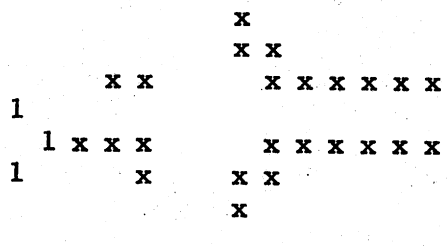
An erasing-wing signal seen entering this element in cycle 0 is seen to create a "12" type signal in cycles 6, 7 and 8. The garbage left blocking the entrance after cycle 11 can be removed (see a later simulation).

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

cycle  1
      x
    x x x
  1     x x x x x x x
    1 x
  1     x x x x x x x
    x x x
      x

cycle  2
      x
    x x x
    1   x x x x x x x
      2
    1   x x x x x x x
    x x x
      x

cycle  3
      x
    1 x x
      2 x x x x x x x

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

cycle  4
      x
      x x
      2 x x x x x x x
      1
      2 x x x x x x x
      x x
      x

cycle  5
        x
        x x
      x   x x x x x x x
        2 1
      x   x x x x x x x
        x x
        x

cycle  6
        x
        x x
      x   x x x x x x x
        1 2 1
      x   x x x x x x x
        x x
        x

cycle  7
        x
        x x
      x 1 x x x x x x x
      1 2   2 1
      x 1 x x x x x x x
        x x
        x

cycle  8
        x
        x x
      1 2 x x x x x x x
      2 x     2 1
      1 2 x x x x x x x
        x x
        x

cycle  9
        x
        x x
    2     x x x x x x x
  x 2 x         2 1
    2     x x x x x x x
        x x
        x

cycle  10
        x
      x x x
    x 1   x x x x x x x
                  2 1
    x 1   x x x x x x x
      x x x
        x

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

cycle  12
        x
        x x
x       x x x x x x x

x       x x x x x x x
        x x
        x

cycle  13
        x
        x x
x       x x x x x x x

x       x x x x x x x
        x x
        x

(END)
```

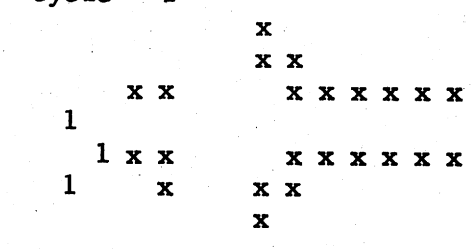
Simulation of the Process of Cleaning-up the Activation Element.

The process of activation in the preceeding simulation leaves some garbage blocking the entrance to the activation element. By the standard construction process discussed earlier, the dirty activation element can be transformed into the configuration in cycle 0. Then an erasing-wing seen entering from the left will destroy this garbage.

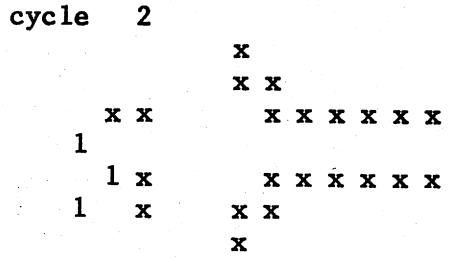
cycle 0



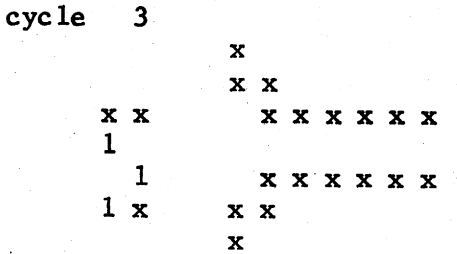
cycle 1



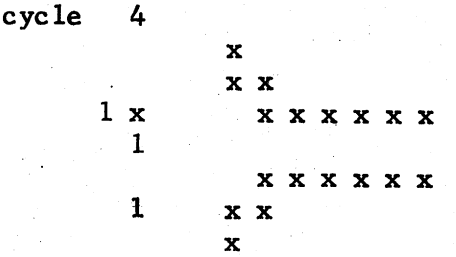
cycle 2



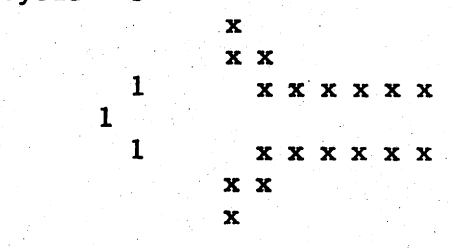
cycle 3



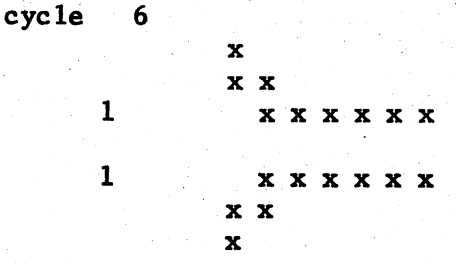
cycle 4



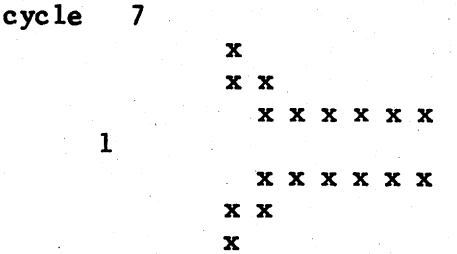
cycle 5



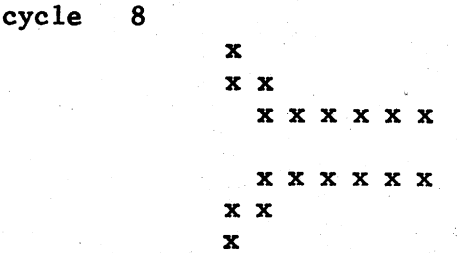
cycle 6



cycle 7



cycle 8



(END)

Biographical Note

The author was born Oct. 2, 1944 in Lebanon, Tennessee, married to Betty Ann Robinson on August 27, 1966. He attended Vanderbilt University from 1962 to 1966 (B.E. in Mechanical Engineering) and the Massachusetts Institute of Technology from 1966 to 1971 (S. M., 1968; M. E., 1970; and Ph. D., 1971 in Mechanical Engineering). A recent publication of the author (Universality in Cellular Automata) appeared in the Eleventh Annual Symposium on Switching and Automata Theory Proceedings.

REFERENCES

1. von Neumann, John, Theory of Self-Reproducing Automata, Edited by A. W. Burks, Univ. of Ill. 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, N. J., 1967.
4. Minsky, Marvin L., and Seymour Papert, Perceptrons, MIT Press, Cambridge, Mass. and London, 1969.
5. Smith, Alvie Ray, III, "Simple Computation Universal Cellular Spaces" in Ninth Annual Symposium on Switching and Automata Theory, IEEE, New York, 1968.
6. Smith, A. R. III, "Cellular Automata Theory", Technical Report No. 2, Digital Systems Labs, Stanford Univ., 1969.
7. Smith, A. R. III, "Cellular Automata and Formal Languages" in Eleventh Annual Symposium on Switching and Automata Theory, IEEE, New York,1970.
8. Hennie, F. C., III, Iterative Arrays of Logical Circuits, jointly by MIT Press and Wiley Sons, Inc., New York and London, 1961.
9. Winograd, Terry, "A Simple Algorithm for Self-Replication", MIT Project MAC Artificial Intelligence Memo No. 197, 1970 and 1967.
10. Banks, Edwin Roger, "Cellular Automata", MIT Project MAC Artificial Intelligence Memo No. 198, 1970.
11. Shoup, Richard G. "Programmable Cellular Logic Arrays", Ph.D. thesis, Carnegie-Mellon Univ., Pittsburg, Pa., March 1970.
12. Beyer, Wendell Terry, "Recognition of Topological Invariants by Iterative Arrays", MIT Project MAC Technical Report No TR-66, 1969.
13. Moore, E. F., "Machine Models of Self-Reproduction", Proc. Symp. Applied Math, XIV, 1961.

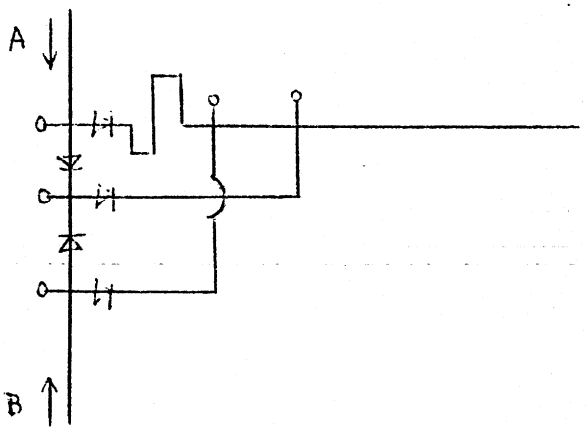
14. Zuse, Konrad, Rechnender Raum, Schriften zur Datenverarbeitung, Vol 1, Freidr. Vieweg and Sohn, Braunschweig, 1969.

15. Amoroso, S., E. Lieblein and H. Yamada, "A Unifying Framework For the Theory of Iterative Arrays of Machines," ACM Symposium on Theory of Computing, May, 1969.

16. Conway, John, "Mathematical Games" (edited by Martin Gardner) Scientific American, Oct., 1970.

Errata

1. page 30, line 2: "The NOT function" should read "The NOR function".
2. page 31, line 1: chapter should be labeled VII, not VI.
3. page 31, third line from bottom: "on time" should be "of time".
4. page 36, third line from bottom: "Similarly the clock, NOT function and NOR function can be constructed." This sentence is not exactly correct. True, the clock and NOT can be constructec as in chapter 6. However, due to the one unit delay of the junction when only a single input is present, the OR as shown there will not work. Although this can be corrected by another configuration, we will correct it by adding two more transition rules, namely (12x2x2) and (212001). The new rules cause the two-input/two-output property of the junction to hold even if one of the inputs is delayed by one time unit. All other components operate as before. The new OR function is shown below. Its operation can be verified by tracing through its operation and remembering that curves (junctions with two dead-ends) also have a delay of one unit.



The OR Function

CS-TR Scanning Project
Document Control Form

Date : 2/8/96

Report # LCS-TR-81

Each of the following should be identified by a checkmark:
Originating Department:

- ☐ Artificial Intellegence Laboratory (AI)
- ☒ Laboratory for Computer Science (LCS)

Document Type:

- ☒ Technical Report (TR)
- ☐ Technical Memo (TM)
- ☐ Other: _____

Document Information

Number of pages: 103 (108 - IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:

- ☒ Single-sided or
- ☐ Double-sided

Intended to be printed as :

- ☐ Single-sided or
- ☒ Double-sided

Print type:

- ☒ Typewriter
- ☐ Offset Press
- ☐ Laser Print
- ☐ InkJet Printer
- ☐ Unknown
- ☐ Other: _____

Check each if included with document:

- ☒ DOD Form
- ☐ Funding Agent Form
- ☐ Cover Page
- ☐ Spine
- ☐ Printers Notes
- ☐ Photo negatives
- ☐ Other: _____

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1 - 103) UNIT'ED TITLE PAGE, 2-36, 36a, 37-101, ERRATA (104-108) SCANCONTROL, DOD, TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 2/8/96 Date Scanned: 2/9/96 Date Returned: 2/15/96

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd

This blank page was inserted to preserve pagination.

UNCLASSIFIED
Security Classification

OUT-OF-PRINT

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Massachusetts Institute of Technology
Project MAC

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP
None

3. REPORT TITLE
Information Processing and Transmission in Cellular Automata

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Ph.D. Thesis, Department of Mechanical Engineering, January 15, 1971

5. AUTHOR(S) *(Last name, first name, initial)*
Banks, Edwin R.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
January 1971	100	16

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-70-A-0362-0002	TR-81 (THESIS)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D.C. 20301

13. ABSTRACT

A cellular automaton is an iterative array of very simple identical information processing machines called cells. Each cell can communicate with its neighbors, and thus on a global basis, the collection of cells is characterized by some type of behavior.

The goal of this investigation was to determine just how simple the individual cells could be while the global behavior achieved some specified criterion of complexity -- usually the ability to perform a computation or to reproduce some pattern.

The chief result described in this thesis is that an array of identical square cells, each cell of which communicates directly with only its four nearest edge neighbors and each of which can exist in only two states, can perform any computation.

A configuration is a specification of the states of all the cells in some area of the iterative array. Another result described in this thesis is the existence of a self-reproducing configuration in an array of four-state cells, a reduction from the previously known eight-state case.

14. KEY WORDS
Cellular Automata Iterative Arrays Universal Computers
Tessellation Automata Parallel Processing Computers

DD FORM 1 NOV 65 1473 (M.I.T.)

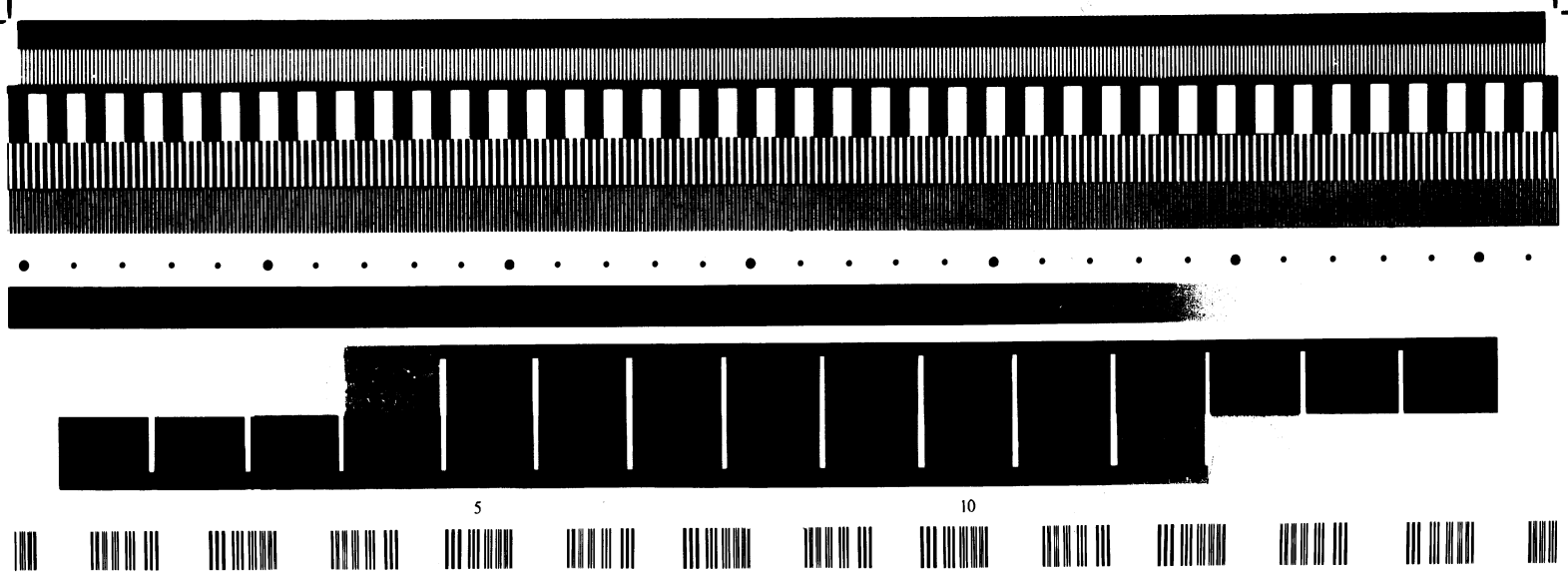
UNCLASSIFIED
Security Classification

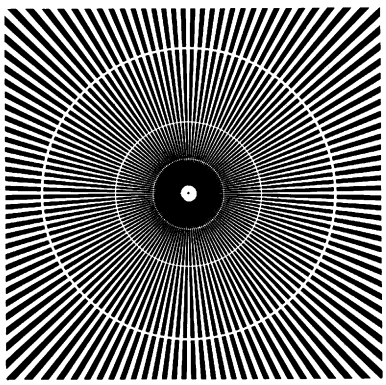
Scanning Agent Identification Target

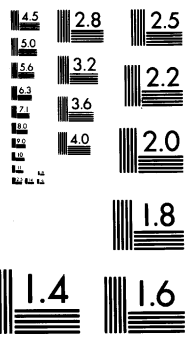
Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned
Date: 2/12/1996
M.I.T. Libraries
Document Services





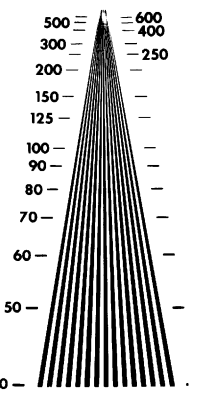


NMA MICROFONT QJKLPYZ
6BS12GH5D4X7U3W8V9E
PQR45DE9UV670FG8STHIJNOWXABYZ
3KLM12C IJK90FGH8STU67AB45CDEZ123VWNOPXYQRLM

ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 OCR-B

ABCDEFGHIJKLMNOPQRSTUV
WXYZabcdefghijklmnopqr
stuvwxyz1234567890PICA

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Elite



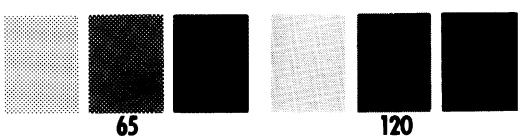
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 6 pt

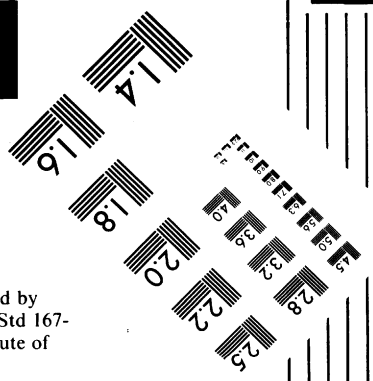
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 4 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 8 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 10 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 12 pt





IEEE Std 167A-1987

FACSIMILE TEST CHART

Prepared by the IEEE Facsimile Subcommittee and printed by Eastman Kodak Company. Use in accordance with IEEE Std 167-1966, Test Procedure for Facsimile. Copyright 1987, Institute of Electrical and Electronics Engineers.