the Moore type. The inputs to each cell are the present states of the four immediate neighboring machines, and its output, therefore, is the present state itself. It follows then that its next-state function is a mapping from the 5-tuple of states of five cells to the states of the cell itself. It is assumed that the operation of all the cells in the space is synchronized. Since there is no input from the outside of the space, von Neumann’s cellular space is an autonomous infinite sequential machine; once an initial distribution of the states of all cells is given, the state distribution of cells in any subsequent time is completely determined. Von Neumann chose a universal Turing machine as a complex and purposeful structure and embedded it in the cellular space as a part of the initial state distribution. Furthermore, his initial configuration embodies a “constructor” which will in time autonomously copy the Turing machine as well as itself, and then the copy makes another copy, and so on. Once the intended goal and the rough idea of the cellular space is described, the remaining task is essentially a synthesis work, and I suspect von Neumann used whatever he saw fit to accomplish his objectives, culminating in the completion of his modeling by the use of a 29-state cell. I think it is a work which has a philosophical impact as well as an artistic value of a kind. To be sure, his work was later improved upon by Thatcher [2], Codd [3], E. Arbib [4], and Smith [5] in various ways.

Lee [6] utilized von Neumann’s cellular space to organize a computer, which seems to be the first attempt to push cellular space beyond self-reproducing automata. Mukhopadhyay’s work is another attempt to utilize von Neumann’s cellular space to synthesize some models other than self-reproducing automata, and he chose to simulate the behavior of finite state machines by modeling them in the cellular space. Since a finite state machine requires an input supplied to it, he had to allow a limited external access to the cellular space, and to designate a specific “output” point, which is also a break-off from the original cellular space concept. Considering the fact that von Neumann could simulate a universal Turing machine and more in the cellular space, it is not surprising to be told that a finite state machine can be simulated in the cellular space. However, one of the differences which exists between these two is that, in the simulation of a Turing machine, the time to be required for the simulation may be extended as much as is needed, while in the simulation of a finite state machine, it is assumed that the simulation of the input–output transition is performed only with a fixed amount of output response delay. The fact that the delay is inevitable in general can readily be seen from the fact that the transmission of any information over n cell distance would require n – 1 time units in the cellular space. In order to make a fixed delay sufficient, Mukhopadhyay makes use of the results of McNaughton’s earlier work [7] on the synthesis of sequential machines based on decision elements which allow limited fan-in fan-out, and which has a fixed nonzero internal delay. Note that a cell in the cellular space is a special case of a decision element with limited fan-in fan-out and with nonzero internal delay. An additional constraint here, above and beyond McNaughton’s, is that the interconnection pattern is also severely restricted. The paper shows step by step how to embed a finite state machine in the cellular space. It is reasonably self-contained and quite readable.

Von Neumann used his cellular space as a scaffold for his model of self-reproducing automata. For this reason, it is rather unfortunate that the theory of uniform arrays of finite state machines and machine self-reproduction seem to have become synonymous to many. In my opinion, however, the modeling of machine self-reproduction is but an example of the uniform array concept even within the modeling of biological processes. Unger [8] found that a two-dimensional array of finite state machines has some interesting processing capabilities for pattern recognition tasks. Bar ricelli [9] has shown, in what is essentially a one-dimensional array, some state-transition phenomena that can be interpreted as analogs of evolutionary processes and natural selection, and Lee and Paull [10] have indicated how a one-dimensional array might be useful for information retrieval systems. All of these are concerned with particular applications in the nature of simulation.

Independent of the above-mentioned efforts, Hennie [11] studied finite iterative arrays from a purely switching theory point of view. Since then, motivated in part by the recent advances in batch fabrication and the integrated circuit technology, there has been a great deal of switching theoretic work concerned with organizing complex logical systems based on finite uniform arrays of switching elements, as Minnick’s survey [12] shows.

The machine array is the underlying structure of all these efforts mentioned. Yet, when it comes to the study of the fundamental properties of machine arrays, their capabilities, and their limitations, surprisingly little has so far been uncovered. Moore [13] and Myhill [14] appear to be the first to have attempted a general study of the concept of machine arrays, and they have not advanced too far. Mukhopadhyay’s work might be considered as an attempt in this direction. The results are neat. Its weakness is that it is still tied down with the specific cellular space of von Neumann. My colleagues and I have for some time been examining various models of machine arrays, and we have made a reasonable abstract model of such arrays and studied it. This is not the place to discuss our model, but our preliminary results [15], [16] indicate that the initial phase of such a study should be of “analytical” nature, as opposed to the “synthetic” approach most of the works discussed herein tend to resort to. It seems the mathematical tools required in the study would range from algebra and geometric number theory to topology, as well as the theory of finite state machines. It is my opinion that an intensive work by many people for a long time yet to come will be required before a unified theory of machine arrays is well established, which seems propitious in the light of its need arising from large-scale integration technology.

Meanwhile, I feel any work, such as Mukhopadhyay’s, which will enrich our knowledge about the properties of general cellular space should be encouraged and welcomed.

H. Yamada
The Moore School of Elec. Engrg.
University of Pennsylvania

REFERENCES

spaces,” University of Michigan, Ann Arbor, Mich., Tech. Rept. 106921-1-T ORA,
1965.
mann,” Bell Telephone Laboratory, Memo. MM63-344-5, 1963.
[7] R. McNaughton, “Badly timed elements and well timed nets.” University of Pennsyl-
1744–1750, October 1958.
Cambridge, Mass.: M.I.T. Press,
1966.
1967.
iterative arrays of machines,” presented at 1969 ACM Symp. on Theory of Computing,
Marina del Rey, Calif.

R70-12 Diagnosis of Single-Gate Failures in Combining Circuits—

This paper is concerned primarily with methods of diagnosis, as opposed with detection of (single-gate) failures in combinational circuits, with but one primary output. Most previous papers in this area have been
concerned with diagnosis and have neglected the detection problem. One result of interest is that it is shown that "for complete diagnosis the number of tests which must be applied to the circuit is proportional to the number of gates and is independent of the number of primary input terminals." Assumed but not given in their method is an algorithm for the purpose of computing a test to distinguish between two failures if such exists.

A gate in a logic circuit is said to be sensitized for some primary input pattern \( T \) if a change in the signal on the gate output causes a change on the primary output. The difficulty here is that for a given failure on the gate, say stuck-at-1, it may be sensitized by \( T \) but for another failure on the same gate, say stuck-at-0, it is not.

A graphical procedure is given, with the above assumption, for computing a set of tests for diagnosing any single-gate failure. It does not require the construction of a fault table. The authors say that a program is being written for this procedure and that they will further investigate the single-gate failure assumption. In LSI such an assumption is probably not valid, but just what would be a reasonable working assumption does not seem to be clear at this time.

J. Paul Roth
IBM Corporation
Yorktown Heights, N. Y.


This paper is based on the following idea. If the two residues \( x_i \cdot f(x) \) and \( x_i \cdot f(x') \) are realizable, respectively, with \( p \) and \( q \) threshold gates, then \( f \) is realizable with at most \( p+q \) gates. And conversely, if the residues require separately at least \( r \) gates, then so does \( f \). Thus, given a table of minimal realizations for 4-argument functions (which require at most three gates), realizations for 5-argument functions can be obtained which are demonstrably minimal or close to it, by considering the five different pairs of residues.

Unfortunately, such tables do not exist for larger \( n \) (and would be very large anyway), so application of the idea for 6-argument functions (and beyond) is difficult. Furthermore, the two bounds on number of gates needed—the worst residue realization \( p \) or \( q \) versus the best sum \( p+q \)—begin to spread apart for higher \( n \), so near-minimality quickly becomes nonguaranteable.

The paper presents an interesting idea well. The reader should be forewarned that positive interconnection weights are assumed.

B. O. Winnder
RCA Laboratories
Princeton, N. J.


The automaton decomposition procedure described by Zeiger does not necessarily yield a unique cascade decomposition of a given machine \( M \). This paper describes some characteristics of the machines in any such decomposition of \( M \) which may be predicted by examining states of \( M \) that are permuted under some input sequence \( x \). The principal results fall into three main categories: cascade consequences of tuples which permutate proper subsets of \( Q_M \), the state set of \( M \); consequences of symbols which permute \( Q_M \); and the decomposition of definite automata.

\( S, \infty \) denotes the largest subset of \( Q_M \) permuted by tape \( x \). It is shown that \( S, \infty \) occurs as a cover element in any sequence of covers \( C_1, C_2, \ldots, C_n \) associated with a decomposition of \( M \). Further, if \( S, \infty \) is an element of \( C_i \) replaced in obtaining \( C_{i+1} \), then each symbol in \( x \) causes a permutation of the states of the corresponding machine of the decomposition. Finally, the machines in any decomposition of \( M \) receive no permutation inputs if and only if each \( S, \infty \) is a singleton.

For an input symbol \( \sigma \) which permutes \( Q_M \), it can be shown that \( \sigma \) permutes the elements of each cover \( C_i \) in any such sequence of covers \( C_1, \ldots, C_n \). The author establishes the following. If \( L_1, \ldots, L_n \) are machines in the cascade decomposition of \( M \) corresponding to \( C_1, \ldots, C_n \), respectively, then \( \sigma \) induces the identity permutation on \( C_i \) if and only if the cascade input \( \sigma \) causes no state change in \( L_{i+1} \).

Finally, definite automata are characterized as precisely those whose cascade decompositions consist of reset machines (ignoring identity permutations resulting from don't-care state transitions).

In some of the proofs, particularly in the early portions of the paper, the complicated notation common in such work tends to obscure essentially simple ideas. The reader will often find a state diagram useful. The author's principal contribution is toward working out some of the consequences of present structure theory. It is to be hoped that this and similar investigations will help shape algebraic machine theory into a tool more useful to the larger community of computer scientists.

W. F. Cutlip
Central Washington State College
Ellensburg, Wash.


This paper deals with the realization of the combinatorial logic network of sequential machines with threshold logic elements, with a view to one-level realization. The state assignment problem is already a difficult one, and here it is further complicated by the restriction that the logical gates to be used are all threshold logic elements. This problem is quite important and is worthy of intensive study because, although threshold logic element as a general gate is well known, its value is not fully appreciated unless its applicability in various situations is demonstrated.

This paper develops an ingenious method of realization, using binary partitions of inputs, states, and outputs. This method is complete in the sense that it starts with a theoretical study and ends with an algorithm yielding the acceptable assignments, although a part of it, which was said to be covered in the Ph.D. dissertation of one of the authors, is not included in this paper. For an assignment to be acceptable, certain conditions must hold for every state partition and every output partition. These conditions, which form the backbone of this paper, are given in Theorem 4 and are rigorously proved.

For convenience and ease of testing, the condition of linear separability is replaced by a slightly weaker condition of 2-summability (complete monotonoticy), which is necessary but not sufficient. This seems to be justified, because for switching functions of eight or fewer variables, 2-summability is known to be a necessary and sufficient condition for a function to be linearly separable.

There are some minor mistakes. It is stated that the addition referred to in domain 2-summability is component-wise Boolean addition. However, to the knowledge of this reviewer, the addition in 2-summability and 4-summability should be ordinary algebraic addition. For instance, let \( t_1 = 1110, t_2 = 1101, t_3 = 1010, \) and \( t_4 = 0101. \) If Boolean addition is used, then \( t_1 + t_2 = 1111 = t_3 + t_4. \) Or the set-pair \( \{t_1, t_2, t_3, t_4\} \) is "equal sum." However, it is easily seen that \( \{t_1, t_2\} \) is linearly separable from \( \{t_3, t_4\}. \)