In Lemma 3, the case that \( t_1 = t_2 \neq t_3 \neq t_4 \neq t_5 \) is missed. In this case, \( \langle t_1, t_2, t_3, t_4 \rangle \) can be either “equal sum” or “not equal sum.”

The study in this paper is not exhaustive in that the situation of multi-level realization is not discussed at all. Suppose it is impossible to obtain one-level realization. What should be the next step? Besides, the criterion of minimization is not clearly specified. If it is allowed to split states, to what extent can state-splitting be done? Is it permissible to achieve one-level realization at the expense of number of variables? In the example given in the paper, there are four output symbols, but three output variables are used instead of a minimum number of two. As stated in the paper, the method is impractical for any but small machines.

In spite of all this, the reviewer considers this paper to be a good contribution in threshold logic as well as in sequential machines, and it also serves as a link between these two fields which have so far not been considered as closely related.

C. L. Sheng
University of Ottawa
Ottawa, Canada


The technology of modern digital circuits displays the following trends.

1) Digital circuits are becoming highly integrated, leading to the technology of LSI (large-scale integration). As a result we note the following.
   a) Delays over large numbers of components, and the consequent timing problems are becoming significant.
   b) As the number of components in the basic module increases, it becomes economically attractive to design the module so as to facilitate fault detection and perhaps fault diagnosis.

2) Individual components are becoming smaller. In fact, the fan-in and fan-out of individual components are both approaching 2. (Hereafter we use “fan” in place of “fan-in” and “fan-out.”)

3) The complexity of each component is being reduced. In particular, the use of components that form a linearly weighted sum followed by a thresholding operation, i.e., “threshold-realizable” components seem to be easier to fabricate than nonthreshold-realizable functions, such as \( \equiv \) and \( \neq \).

Techniques for coping with one or two of these trends have been described in some papers [1]–[5] but no single paper has yet described a technique for coping with all of these trends. In particular, in recent years, an abundance of theoretical work has appeared on 1) threshold logic [1] and 2) low-fan cellular arrays [2]. This is the first paper, to my knowledge, that treats the theoretical aspects of low-fan threshold-logic cellular realizations of arbitrary switching functions. It may be that this combination of threshold logic and cellular arrays will make both threshold logic and cellular arrays much more economically attractive.

Combining the concepts of threshold logic and cellular arrays in this paper has resulted in a happy marriage of practical potential and theoretical power. Many basic properties of threshold functions\(^1\) have been specialized to two inputs and to cascade arrays, yielding a useful specialized theory.

The main result of the paper is an algorithm (Algorithm 2) that realizes an arbitrary switching function by a two-dimensional unate cellular array, and that often achieves this by a minimal array. The paper also includes an algebraic test-realization procedure for unate cascades, as well as methods for realizing minimal arrays when the ordering of the inputs is specified.

The theory as it stands in this paper has the shortcoming of requiring a specific ordering of the variables in the synthesis of minimal-array realizations of arbitrary switching functions. Beyond this shortcoming, we hope to see, in future papers on this subject, theories leading to the realization of threshold cellular arrays that 1) have so many elements that the overall delays are significant, and 2) lend themselves to fault detection and diagnosis.

J. Sklansky
University of California
Irvine, Calif.

REFERENCES


\(^1\) Every two-input threshold function is unate, and vice versa [1]. Hence the author's use of "unate" for two-input gates is equivalent to "threshold."