Reviews of Papers in the Computer Field

DONALD L. EPLEY, Paper Reviews Editor
D. W. FIFE, A. I. RUBIN, R. A. SHORT, H. S. STONE
Assistant Paper Reviews Editors

Please address your comments and suggestions to the Paper Reviews Editor: Donald L. Epley,
Department of Electrical Engineering, University of Iowa, Iowa City, Iowa 52240.

A. SWITCHING THEORY


The concepts of homomorphism and substitution property have played an important role in the structure theory of complete deterministic automata. In this paper the author tries to extend them to the general case of incomplete nondeterministic automata. 1

An automaton M is a triple (S, A, δ), where S is a nonempty set (of states), A is a finite set (of input symbols), and δ is the transition relation. It is convenient to introduce the notation δ_a = {(s, t) | (s, a) ∈ δ, t ∈ S} for each a ∈ S, where s, t ∈ S. Two automata M = (S, A, δ) and M = (T, A, δ) are fixed throughout the paper.

A relation φ ⊆ S × T is called a generalized homomorphism (GH) from M to N if (v ∈ A) [(v) δ_a ⊆ φ] 2 A GH φ from M to itself is called a generalized congruence relation (GCR), if it is reflexive and symmetric. In this case φ satisfies (v ∈ A) [(v) δ_a ⊆ φ], which is called the substitution property. A cover with substitution property (or SP cover) is used in the usual sense. It is shown that every GCR φ on M defines an SP cover C_φ on M by

C_φ = {R ∈ S | (v) ∈ R}.

Define δ_φ by

(v) δ_φ = {C(v) C(v) ∈ C_φ | C(v) ⊆ φ C_φ = φ}.

where φ is the empty set. Then it is shown that every GCR φ on M determines a unique GH φ(φ(0)) from M to M/θC_φ, much like in the complete deterministic case. It is also shown that if φ is a GH from M to a complete automaton N such that I_a = φ δ_0 = φ, where I_a = {s, t} | (s, t) ∈ S}, then φ determines a unique GCR φ(θ) = φ δ_0 = φ, on M.

A generalized isomorphism between M and N is defined to be a GH φ ⊆ S × T from M to N such that φ = φ δ_0 S ⊆ φ, φ δ_0 T ⊆ φ, I_a = φ δ_0 = φ. It is then shown that the generalized isomorphism is indeed a generalized notion of isomorphism.

According to the author’s definition the empty relation φ is a GH, which appears to be too far-fetched a generalization. Lyndon’s definition 2 of homomorphism as cited by the author has the same difficulty, but it is


2 Let X, Y, and Z be arbitrary sets, X Y, Y Z, Y Z, and S Y Z. The composition “ο” is defined by θo = ((x, y) | (x, θ(y) ∈ S)}, X = (x | (x, y) ∈ X)}, Y = (y | (x, y) ∈ Y)). The relation “ο” is defined by θ | (x, y) ∈ X, y ∈ Z). The relation “ο” is defined by θ | (x, y) ∈ X, y ∈ Z).

This paper suffers from an unusually large number of typographical errors and some of them are quite intriguing. This reviewer could not figure out what M_φ (θ) was, until the author told him that it should be M_φ (θ).

T. KAMEDA
University of Waterloo
Waterloo, Ontario
Canada


One of the most important tasks of a present day design automation system is the packaging of logic networks. Typically this process follows these steps:

1) partitioning the logic network into modules subject to the packaging constraints;
2) placement of the modules on a chip or printed circuit board;
3) routing of the conductors necessary to properly interconnect the modules.

Each of these steps has produced a number of interesting problems due to the many packaging technologies involved and the variety of constraints which often must be imposed.

This paper addresses the problem posed by the first step—that of partitioning (or clustering) the N gates of a logic network into a set of discrete modules which can then be placed and interconnected. There are many different constraints which may have to be observed in the selection
of these modules, but two are nearly always present: 1) no more than $M$ gates shall appear in a single module, and 2) no module shall require more than $P$ pins. Since it is normally not too difficult to discover a set of modules meeting these constraints, the important and challenging part of the problem is that of making such a selection so that a prespecified objective function is optimized. One common goal, for example, is that of simply minimizing the total number of modules; another is to minimize the number of wires required to interconnect the modules. Several papers have appeared which attack these and similar problems involving such quantitative objective functions.

In this paper the authors consider a slightly different but equally important question: How can the modules be chosen so that the maximum delay through the network is minimized? For this purpose they assume that no delay will be encountered within a module, but that a delay of one unit will occur between modules. They also make the initial assumption that the given logic network is acyclic, i.e., that no signal can feed back on itself, and that the requirement of a maximum of $M$ gates and $P$ pins per module must be observed.

While the complete algorithm which they present is reasonably complex, they proceed to develop it in a series of steps which individually can be easily followed. They begin by neglecting the $P$ pin constraint and assuming that the given network is a rooted tree. (A rooted tree results when all the branches of a tree are directed toward (away from) a single node. It is also known as an arborescence.) They solve this first problem of clustering the gates of this rooted tree so as to minimize the maximum delay by means of a node-labeling procedure quite similar to that normally used to find the maximum delay through an arbitrary acyclic network. However, because of the $M$ gate constraint on the modules, the procedure is modified to ensure that this constraint is not violated.

The authors then proceed to nonrooted trees where their key result is the fact that for any nonrooted tree there exists a set of branches whose removal leaves a set of rooted trees. Each of these rooted trees can then be clustered by the previous procedure and its maximum delay will be minimized. Clearly, the maximum of these individual delays gives a lower bound on the maximum delay for the complete network. Now, however, when the set of edges which was initially removed is restored, complications may arise. This results from the fact that new paths containing these restored edges now appear and one or more of their delays may be greater than that for any of the individual rooted trees. If, however, both of the endpoints of a restored edge can be relocated in the same cluster, then (by the initial assumption) the delay for this edge becomes zero, thus reducing the delay of all paths through it by one. The authors give a systematic procedure for reclustering and relabeling such edges so that the optimal labeling (and hence optimal delay) is ultimately achieved. A detailed example of the complete procedure for nonrooted trees is included.

At this stage in the presentation the obvious next step is to drop the “tree restriction” and consider acyclic graphs in general. (In particular, acyclic graphs whose underlying undirected graphs contain closed loops.) Here, however, the authors claim to “have been unsuccessful in developing an efficient algorithm.” Since the procedures that have been developed up to this point are quite straightforward and logical, one tends to wonder if perhaps the authors’ definition of “an efficient algorithm” was set too high. The general problem seems to be of sufficient importance and difficulty so that even a “suboptimal” solution would have been welcomed.

The problem which the authors do consider is that of clustering an arbitrary acyclic network where “node replication” is permitted. By this they mean the option of introducing identical copies of various logic gates and supplying each of them with the same set of input signals. A labeling algorithm is then described and illustrated which again achieves the optimal delay. Once the network that achieves this delay has been found, it is often possible to remove many of the replicated nodes and a procedure for accomplishing this is described. The observation is made that while the concept of “node replication” does not provide a general solution to the problem of arbitrary acyclic networks, by permitting it the designer can often achieve a smaller delay than would otherwise be possible.

The final technique described involves the $P$ pins per module constraint which was initially neglected. It turns out that each of the three algorithms is easily modified to accommodate this requirement and the appropriate modifications are described. It is noted that the number of computations for each of the algorithms varies with $N^2$. The paper concludes with a list of unsolved problems and areas for generalization.

The paper is relatively easy to understand and follow, not because the algorithms are particularly simple, but because the presentation is clearly and logically organized. An algorithm (even if not optimal) for the general case would have been useful, but this does not detract from the significance of the techniques which are developed. Certainly they form a firm structure upon which a more general solution can be developed.

Sheldon B. Akers, Jr. General Electric Co. Syracuse, N. Y.


This is an interesting paper. The authors consider two problems related to the testing of universal cellular tree circuits. The first is concerned with the detection and location of faulty cells by applying a minimal number of tests. The second is concerned with that of finding a set of switching functions that can be implemented by a faulty tree circuit.

A cellular tree circuit (with $n$ levels), capable of implementing any function of the $n$ variables $x_1, x_2, \ldots, x_n$ is made up of $2^n - 1$ cells where each cell $C_j$ is defined by

$$C_j = C_{j-1}^2 x_i + C_{j-1}^2 x_i$$

for $i = 1, 2, \ldots, n$, and $j = 0, 1, \ldots, 2^n - 1$. The binary inputs $C_0, C_1, C_2, \ldots, C_{2^n - 1}$ are variables for specifying the function $F(x_1, \ldots, x_n)$ of the circuit.

The assumed failure modes are such that the output of a faulty cell $C_j$ remains a function of its three inputs $x_i, C_{j+1}^2$, and $C_{j-1}^2$. There are four sets of tests the authors considered: $O_0, O_1, O_0, \text{ and } O_1$. The $O_j$ set is composed of those tests where each $C_j$ (for $0 \leq j \leq 2^n - 1$) is fixed at 0 and the inputs $x_1, \ldots, x_n$ take on all possible variations of $2^n$ patterns. This is the so-called “fixed-0 test.” The $O_1$ set is composed of those tests where (for $C_0, C_{2^n-1}$) taken on $(011 \cdots 1), (101 \cdots 1), \ldots, (1 \cdots 1)$, while the inputs $(x_1, \ldots, x_n)$ take on $(00 \cdots 0), (10 \cdots 0), (01 \cdots 1)$. This is the so-called “traveling-0 test.” The $O_0$ and $O_1$ sets are then composed of those obtained by complementing, respectively, the $O_0$ and $O_1$ tests. The whole test set $(O_0, O_1, 0, 1)$ therefore contains $2^n + 2$ tests, each $2^n$ bits long.

In the first part of the discussion, the authors prove that the necessary and sufficient conditions for an $n$-level tree circuit to function correctly is that the tests $(O_0, O_1, O_0, 1)$ give the correct outputs (i.e., four sequences of 0’s, 1’s, 0’s, 1’s, respectively, each of $2^n$ bits). Although there is no mention of programmed testing time, it appears that the method is easily programmable and a small circuit, say of 12 variables or less, can be tested within a reasonable amount of time.

In the second part of the discussion, the authors point out that the test results obtained by applying the tests $(0, 1, 0, 1)$ do not yield good diagnostic resolution. The additional tests required for fault location may become prohibitive. An alternate approach was therefore to utilize the faulty tree circuit, by analyzing the test results of $(0, 1, 0, 1)$, to determine the functions that can be implemented. This is a rather involved process; it does not appear attractive for practical size circuits.

This reviewer feels that the problem of testing large-scale arrays is becoming more and more acute because of the low yield and the combinatorial nature of test patterns. This situation is worsened in the case of universal cellular arrays. Perhaps a more realistic way to approach the problem is to realize cells with only the specialized switching functions (such as and-