leads running across the circuits to bonding pads on either end of the substrate, and with a common gate circuit which may be either external to the substrate or mounted on it.

Fig. 9. A 12-detector substrate. The substrate is glass (0.8 by 1.2 inch) with 218 tantalum resistors and 144 transistors.

Conclusion

This paper has described a digit detector suitable for wide application in memory systems. The circuit can be dc or ac coupled to a digit line, and can be used with two different digit drive arrangements. Connection can be made at the sending, receiving, or middle of a digit line. It is suitable for a low-cost, integrable realization.

Acknowledgment

The author is indebted to T. R. Finch, S. Waaben (co-inventor), K. D. Wise, and W. C. Slemmer for their encouragement, advice, and helpful suggestions.

References


Module Clustering to Minimize Delay in Digital Networks

EUGENE L. LAWLER, MEMBER, IEEE, KARL N. LEVITT, MEMBER, IEEE, AND JAMES TURNER

Abstract—An important aspect of the packaging of digital networks is the allocation of logic gates to modules such that a predetermined objective function is minimized. In order to develop techniques for this partitioning of a logic network we have considered the following problem: Given an acyclic combinational network composed of various primitive blocks such as NOR gates, assume that a maximum of \( M \) gates can be "clustered" together into larger modules, and that a maximum of \( P \) pins can be accommodated in each larger module. Assume also that in a network composed of such larger modules, no delay is encountered on the interconnections linking two gates internal to a module and a delay of one time unit is encountered on interconnections linking two gates in different modules. Find an easily applied algorithm that will result in a network such that the maximum delay through the network is minimized. Efficient, easily applied algorithms for optimal clustering are presented for the case in which the combinational network is in the form of a tree. For the more general case in which the combinational network is represented by a directed acyclic graph which is not a tree, an optimal module clustering is obtained, but subject to the provision that the primitive blocks of the network may be replicated.

Index Terms—Graph decomposition, logic partitioning, minimization of longest delay.

I. Introduction

In recent years there has been extensive interest in the problem of the placement of modules of an arbitrary network on a plane and the routing of backplane intermodule wires in order to minimize such objective functions as the total wire length, the wire length in the longest closed path, or the wire length in the longest path from a network source to a network sink. Breuer [1] has compiled a comprehensive survey of the results on this so-called "placement-routing"
problem, which at present does not admit of a computationally satisfactory optimum solution, although several heuristic solutions have been formulated [2].

The state of affairs has been considerably complicated in recent years by advances in semiconductor technology. If initially a network of simple gates is given—say 3-input NOR gates—it is reasonable to assume that somewhere between 5 to 1000 gates (as measured by how far one wishes to extrapolate the technological capability) can reside together on a single semiconductor chip. Such a cluster of gates can then form a module in the "placement-routing" problem.

In some cases—and it may become increasingly appropriate in large-array design—the module designs are obtained by partitioning a large logic network into clusters of gates that satisfy packaging constraints. The choice of partitions, or clusters, clearly affects the number, length, etc., of intermodule connections. It is unreasonable to expect that the resulting combined "clustering-placement-routing" problem would be amenable to an efficient optimum solution, but from a computational viewpoint, it appears that there is some advantage in seeking computationally efficient heuristic solutions to the problem, or, at least, in decomposing the problem into two or three parts. In this paper we are considering techniques for clustering 1 gates of a combinational network. There are numerous criteria that might be of interest in such clustering. In this case, we seek to minimize a measure of the delay through the network. The basic problem is the following.

Given an acyclic 2 combinational network, composed of various primitive blocks such as NOR gates, etc., assume that a maximum of $M$ gates can be accommodated in larger modules, and that a maximum of $P$ pins can be accommodated in each larger module. Assume also that in a resultant network composed of such larger modules a) no delay is encountered on an interconnection linking two gates internal to a module, and b) a delay of one time unit is encountered on an interconnection linking two gates in different modules. Find an easily applied algorithm that will result in a network such that the maximum delay through the network is minimized.

The assumption of negligible intramodule delay compared with the intermodule delay appears to be quite valid, as evidenced by the subnanosecond gate-propagation times predicted for the near future by semiconductor technologists.

In Section II we develop the graph-theoretical preliminaries pertinent to the problem. In Section III, IV, and V we consider, respectively, for the $M$-gate accommodation constraint only, optimal clustering techniques for rooted tree networks, nonrooted tree networks, and general combinational networks with reconverging fan-in. In Section VI an algorithm for optimal clustering in the general case with the pin limitation is developed. Not surprisingly, as shown in [4], ch. 2, the problem can be cast as an integer linear program, but the solution developed is computationally inefficient since the numbers of variables and constraints are both proportional to $N^2$, where $N$ is the number of gates in the network.

II. Graph-Theoretical Preliminaries

We represent a combinational network as an acyclic directed linear graph $G$. We will be considering separately graphs in the form of "rooted trees," "nonrooted trees," and finally, "acyclic graphs," which represent networks of increasing generality.

1) An acyclic graph is a directed graph that contains no directed closed paths. Minimal gate realizations of functions will usually assume the form of acyclic graphs which are not trees.

2) A nonrooted tree is an acyclic directed graph such that the underlying undirected graph is connected and contains no closed paths. Such a tree can represent the network form for multiple-output function decomposition.

3) A rooted tree is a directed graph with only a single node with out-degree zero (a "sink") or only a single node with in-degree zero (a "source"), and such that the underlying undirected graph is connected and contains no closed paths (cycles). If the tree contains a single node with out-degree zero, we call the tree a "sink tree"; otherwise, the tree is called a "source tree." A sink tree can represent a network with a single output.

Node $i$ is said to be a "predecessor" ("successor") of node $j$ if there exists a directed path from node $i$ to node $j$ (from node $j$ to node $i$). The set of predecessors of a set $A$ is the set of all nodes which are predecessors of at least one node in $A$.

Before describing efficient clustering algorithms, it is of interest to investigate upper bounds on the computation complexity if an exhaustive approach is assumed to find the optimum solution. As a first approximation to a lower limit for the upper bound, let us consider the number $C_1(N, M)$ of distinct ways of clustering the nodes of a graph of $N$ nodes into groups containing a maximum of $M$ nodes. It can be shown [4] that

$$C_1(N, M) > \alpha N^{\beta N},$$

where $\alpha, \beta$ are independent of $N$, indicating that if an exhaustive approach is utilized, the computation growth rate with $N$ is in excess of $N!$.

The above upper bound can be made significantly tighter by noting that for the minimization of the longest network delay, only those clustering arrangements need be considered wherein "neighboring" nodes are clustered. The following lemma, stated without proof, is a reflection of this observation.

Lemma 1: There exists an optimal clustering arrangement such that the "restriction" of the graph to the

---

1 Lawler [3] has studied the somewhat related problem of clustering the gates of a network (not necessarily combinational) into a set of larger modules which are limited in the number of gates and external pins that can be accommodated, with the objective of minimizing connections between modules.

2 The authors' colleague, W. H. Kautz, and the reviewer have noted that some minimal gate-type networks have acyclic form, and yet exhibit only combinational behavior at the outputs. However, the following discussion will only be concerned with acyclic networks.
nodes of any cluster, i.e., the graph formed by the nodes of the cluster, and all edges connecting those nodes, is a connected graph (when regarded as an undirected graph).

Applying Lemma 1, it can be shown [4] that the number of the clustering arrangements, \( C_3(N, M) \) which need to be investigated in order to reveal an optimal clustering is given by \( C_3(N, M) > \delta^3 N \), where \( \delta \) is the maximum out-degree of a node in a graph, and \( \delta \) and \( \lambda \) are constants independent of \( N \). Hence, in order to determine the optimum clustering arrangement, using a "selective" exhaustive technique, a number of computations is required that increases at least exponentially with \( N \). In the following sections we will describe clustering algorithms for which the computation growth does not exceed \( MN^2 \).

For the case where the only clustering constraint is concerned with the maximum number of gates which a larger module can accommodate, we can conveniently formulate the problem of finding an efficient algorithm as that of labeling the nodes of the graph with integers.

The label given to a node will indicate the maximum delay along any path from a source of the network to the node in question. In labeling the nodes, for all three directed graph types, we will observe the following rules.

**Rule 1:** Nodes with in-degree zero (sources or, equivalently, input nodes) will be given the label 0.

**Rule 2:** A node is to be given a label at least as large as the largest label given to any of its predecessors.

**Rule 3:** For each label \( k \) let \( G_k \) denote the restriction of the graph to nodes bearing label \( k \), i.e., \( G_k \) is the subgraph formed by all edges joining nodes with label \( k \). Each connected component of \( G_k \) is to contain \( M \) or fewer nodes.

We will seek algorithms such that the above rules are satisfied, and such that the largest label is as small as possible. The largest label denotes the maximum delay through the network.

### III. Labeling Procedure for Rooted Trees

It turns out that there are advantages in considering a mild generalization of the clustering problem. Associated with each node \( i \) of the graph, in addition to its label, we shall assume there is a nonnegative weight \( w_i \). We shall then demand that the total weight of all nodes in a cluster not exceed \( M \). Thus, Rule 3 as stated above will read "... Each component of \( G_k \) is to contain nodes, the sum of whose weights does not exceed \( M \)."

Now it is apparent that in the case of rooted trees that are sink trees, each component of the subgraph \( G_k \) mentioned in Rule 3 will also be a sink tree. Hence, for the special case of sink trees, Rule 3 can be replaced by the following more easily applied rule.

\[ \text{Rule 3': The sum of the weights of any given node, and all of its predecessors which have the same label, is not to exceed } M. \]

Suppose the largest label assigned to any of the predecessors of node \( i \) is \( k \). We call all such predecessors having this label \( k \) the "\( k \)-predecessors" of node \( i \). The total weight of all the \( k \)-predecessors of node \( i \) will be denoted \( w_i(k) \).

The following simple labeling procedure for sink trees is seen to minimize the largest label.

**Step 0.** Label all source nodes 0.

**Step 1:** Find any unlabeled node \( i \), all of whose predecessors have been labeled. Let \( k \) be the largest label applied to any of these predecessor nodes. If \( w_i + w_i(k) \leq M \), then apply the label \( k \) to node \( i \); otherwise, apply the label \( k+1 \).

Repeat Step 1 until all nodes are labeled.

The connected restrictions of the graph form the clusters.

The application of this labeling procedure to a sink tree for \( M=3 \), assuming the weight of all nodes is unity, is shown in Fig. 1, where the labels are shown enclosed in parentheses. Lines are drawn around nodes in the same cluster. It is trivially noted that the number of resultant clusters can be reduced without exceeding the node accommodation restriction, and without increasing the longest path, by combining into single clusters nodes 3, 4, and 5, nodes 7, 10, and 12, and nodes 8, 9, and 11. Lawler's method [3] can be utilized in less trivial cases to form a minimal cluster set from the cluster set which results from the labeling.

We note that this labeling procedure can equally well be applied to source trees, by substituting "sink" for "source" in Step 0 and substituting "successors" for "predecessors" in Step 1. In the following sections we shall generally use primes to refer to the labels and weights of successor nodes. Thus, \( k' \) will denote the largest label assigned to any successor node of node \( i \), and \( w'_i(k') \) will denote the sum of the weights of all \( k' \)-successors.

Formal proofs will not be presented for the optimality of the clustering which results from this algorithm, or from succeeding algorithms. However, it should be
emphasized that the clusterings are indeed optimal, and that the proofs follow immediately from the algorithm statements.

IV. LABELING PROCEDURE FOR NONROOTED TREES

A. Separation of Nonrooted Trees into Rooted Trees

In this section we will develop a procedure that will minimize the longest delay in nonrooted tree networks; the labeling procedure will rely upon Lemma 2, and its sequel, which describes a procedure for separating a nonrooted tree into a set of rooted trees. Once the separation process has been completed, each resultant rooted tree is labeled according to the procedure outlined in Section III; certain systematic modifications may then be required to yield an optimum solution.

Lemma 2: For any nonrooted tree, there exists a set of edges $A$ such that a) removal of these edges separates the nonrooted tree into a number of rooted trees, and b) each source-sink path in the nonrooted tree passes through at most one of these edges.

The proof of this lemma is easily established by the following constructive procedure for the separation.

Step 0: Choose any source node and all of its successors as an initial source tree. Place a check mark on all of the nodes in this tree.

Step 1: Find an edge, exactly one end of which is checked, and place this edge in $A$. If the checked end of this edge is the head (tail), then the node at the tail (head) and all of its predecessors (successors) is chosen as a sink (source) tree. Place a check mark on all nodes of this tree.

Repeat Step 1 as often as necessary.

In Fig. 2 we illustrate the tree separation process for a nonrooted tree, choosing arbitrarily node $a$ as the initial source node; the shaded edges represent members of the set $A$. By construction we note that the nonrooted tree is clearly separated into rooted trees, thus complying with a) of Lemma 2. Since each edge in $A$ connects a sink tree to a source tree, at most one edge in $A$ is encountered in any source-sink path, in accordance with b) of Lemma 2.

Another way to visualize the separation is by means of the schematic of Fig. 3, in which the rooted trees are numbered according to the order in which they are constructed. Although this order is not unique, it plays an essential role in the clustering procedure.

B. Initial Phase of Algorithm

The labeling procedure for nonrooted trees is conveniently described in terms of an initial phase and a final phase. For the initial phase we have the following steps.

Steps 0 and 1: The tree is separated into rooted trees as described above.

Step 2: Each rooted tree is labeled by the procedure described in Section III. It is recalled that the labeling is taken from "sinks to source" in the case of source trees, and from "source to sink" in the case of sink trees.

Step 3: Calculate

\[ D = \max_{i \in N} \{ k_i \} \]

and

\[ D' = \max_{(i,j) \in A} \{ k_i + k_j + 1 \}, \]

where $k_i$ and $k_j$ are the labels applied to nodes $i$ and $j$, $(i, j)$ is the edge connecting nodes $i$ and $j$, and $N$ denotes the entire set of nodes of the nonrooted tree. If $D' \leq D$, proceed to Step 6 (see Section IV-C). If $D' \geq D + 1$, proceed to the relabeling procedure, Step 4.

An explanation of these steps is as follows. The labels assigned in Step 2 imply a feasible clustering of the nodes, for which the nodes at either end of any edge in set $A$ are assumed to be in separate clusters. Any source-sink path is either a) entirely contained within one of
the separate rooted trees, or b) starts in a sink tree, passes through exactly one edge in the set $A$, and ends in a source tree. The maximum delay encountered along a path of type a) is

$$D = \max_{i \in M} \{ k_i \},$$

i.e., the maximum label assigned to any node. The maximum delay encountered along a path of type b) is

$$D' = \max_{(i,j) \in A} \{ k_i + k_j + 1 \}. $$

If $D' \leq D$, then the clustering is optimal, since it is not possible to make the delay for the entire network less than the delay for an optimal clustering of one part, i.e., a single rooted tree.

On the other hand, if $D' \geq D+1$, there are one or more edges $(i,j) \in A$ which are "critical," in the sense that

$$D' = k_i + k_j + 1 \geq D + 1.$$ 

It may be possible to reduce the maximum delay—by one unit only—by placing both ends of all such critical edges in the same cluster. However, in the process of doing so labels may be changed for various nodes, and certain other edges may therefore become critical. There is a systematic way to carry out a computation for placing critical edges within clusters, which is described next.

C. Final Phase of Algorithm

In the final phase of the algorithm, we attempt to relabel the nodes of the nonrooted tree in such a way that the maximum delay is reduced to $D' - 1$. In doing so, we shall relabel each of the separate rooted trees in the opposite order from that in which they were formed. (Recall that we commented that this order plays an essential role in the algorithm.) The effect of relabeling the rooted trees in this order is to insure that when we are relabeling a source (sink) tree, the tails (heads) of all edges in the set $A$ which are directed into (out of) the rooted tree in question will already have been relabeled (except for the edge incident to the root of the tree). This allows us to have available all of the information which is necessary for relabeling each successive node.

Suppose we are to relabel node $i$ of a sink tree, where node $i$ is any node other than a source or the sink (obvious modifications are made for source trees). Having already relabeled all predecessors of $i$, let $k$ be the largest of these new labels, and let $k'$ be the largest label of a successor of $i$ lying outside the sink tree in question, i.e., of course, contained in a source tree. We must observe the following principles.

1) If $k + k' > D' - 1$, then the delay clearly cannot be reduced to $D' - 1$, no matter how we cluster the node $i$.
2) If $k + k' = D' - 1$, then if the delay is to be reduced to $D' - 1$, node $i$ must be given label $k$ and clustered with all its $k$-predecessors and $k'$-successors lying outside the sink tree in question. This is possible if and only if $w_i + w_i(k) + w_i(k') \leq M$.
3) If $k + k' < D' - 1$, then in the first labeling no edge from $i$ in the set $A$ is critical, and we now give $i$ label $k$ if $w_i + w_i(k) \leq M$.
4) If in 3) we find that $w_i + w_i(k) > M$, node $i$ will have to have label $k+1$, and we must consider then whether any edge from $i$ (connecting the sink tree in question to a source tree) has become critical. If none have become critical, i.e., if $(k+1) + k' < D' - 1$, then continue on.
5) If in 4) $(k+1) + k' = D' - 1$, then an edge from $i$ became critical when label $k+1$ was given to node $i$.
   In this case, we must test whether or not it is possible to cluster node $i$ with its $k'$ successors lying outside the tree in question, i.e., test whether $w_i + w_i(k') \leq M$. If this test fails, then a delay of $D' - 1$ cannot be achieved.
6) In 2) and 5), when node $i$ is clustered with its $k'$-successors, the weight $w_i$ must be augmented by $w_i(k')$, so that later, when considering some successor of $i$ in the tree in question, one has an accurate determination of the number of nodes already bearing the same label as $i$.

Table I summarizes this information for relabeling sink trees. A similar table could be given for source trees by simply interchanging everywhere they appear the letters $k$ and $k'$ and the values $w_i(k)$ and $w_i(k')$, i.e., $k'$ would refer to the label of node $i$ of a source tree node, $k$ to the label of the predecessor of node $i$ in a sink tree, $w_i(k')$ to the total weight of all of the successors of node $i$ with label $k$, and $w_i(k)$ to the total weight of the predecessors of node $i$ (in the sink tree) with label $k$.

Notes for Table I:

1) Even if it were possible to place node $i$ in the same cluster as its $k$-predecessors and $k'$-successors, this would not allow delay to be reduced to $D' - 1$.
2) Node $i$ must be placed in the same cluster as its $k$-predecessors and $k'$-successors if delay is to be reduced to $D' - 1$.
3) Node $i$ cannot be placed in the same cluster as its $k$-predecessors because of weight limitations.
4) Node $i$ cannot be placed in the same cluster as its $k'$-successors because of weight limitations.
5) It is not necessary to place node $i$ in the same cluster as its $k'$-successors because delay is not critical.
6) Node $i$ must be placed in the same cluster as its $k'$-successors if delay is to be reduced to $D' - 1$.

An example may illustrate various points of detail in Table I (an example of the full algorithm is given in
Section IV-D). The numbers beside nodes in Fig. 4 are labels; the shaded edges represent edges in A. Previously determined clusters are outlined. All node weights are assumed to be unity, unless otherwise stated. We assumed \( D' - 1 = 6 \) and \( M = 4 \).

The largest label assigned to any of the predecessors of node \( i \) is \( k = 2 \). There are two such 2-predecessors, one of which has its 4-successors in a separate rooted tree placed in the same cluster. Assuming the weight of all nodes was originally unity, the weight of the 2-predecessor in question has been augmented by 3 to 4. Thus, \( w_i(2) = 5 \), and if \( M = 4 \), we see that node \( i \) cannot be placed in the same cluster as its 2-predecessors. The largest label applied to any of the successors of node \( i \) in other rooted trees is \( k' = 3 \), and \( w_i'(3) = 4 \) because the weight of one of these 3-successors has been augmented previously, and node \( i \) cannot be placed in the same cluster as its \( k' \)-successors. Thus, \( k + 1 = 3 \) is the smallest label that can be assigned to node \( i \), and the total delay will be \((k+1) + k' + 1 = 7\). Hence, no reclustering will reduce the delay below that achieved by the initial clustering.

We can now summarize the final phase of the algorithm.

Repeat Step 4 below for each rooted tree in turn, taking the rooted trees in the opposite order from that in which they were formed in Steps 0 and 1.

**Step 4:** Assume the tree is a sink tree (modification of the procedure for source trees is obvious).

a) Label all source nodes 0.

b) Find any unlabeled node \( i \), all of whose predecessors have been relabeled. If this node is the sink node, pass to substep c) below. Let \( k \) be the largest label of any predecessor node, and \( k' \) the largest label of any successor node (not contained in the sink tree in question). Assign a new label to node \( i \) according to Table I. Repeat this step as often as necessary.

c) In the case of the root node, let \( k \) be the largest label of any predecessor node. If \( w_i + w_i(k) \leq M \), assign the label \( k \) to the node; otherwise, assign the label \( k + 1 \). If the value of this label is greater than \( D' - 1 \), go to Step 6.

**Step 5:** Replace each label \( k_i \) obtained for a source tree in Step 4 by \((D' - 1) - k_i \); the labels for sink trees remain the same as determined in Step 4. The resulting labeling is optimal. End the procedure.

**Step 6:** Replace each label \( k_i \) obtained for a source tree in Step 2 by \( D - k_i \); the labels for sink trees remain the same as determined in Step 2. The resulting labeling is optimal. End the procedure.

D. Example

An example of the complete application of the algorithm is shown in Fig. 5. In Fig. 5(a), the nonrooted tree is decomposed into rooted trees in the order indicated by the parenthesized numerals (Steps 0 and 1). In Fig. 5(b), each of the rooted trees is labeled (step 2), assuming \( M = 3 \). Upon application of Step 3, it is seen that \( D = 3 \) and \( D' = 5 \). Figure 5(c) shows the result of relabeling. During the relabeling process, no real changes are made until the node \( i_1 \) of tree 0 is encountered. For \( i_1 \), we have \( k' = 1, k = 2, w_i'(k') = 3 \), and \( w_i(k) = 1 \). Since \((k' + 1) + k = 4 = D' - 1\), the portion of Table I (modified for a source tree) associated with note 6 applies. Since \( w_i + w_i'(k) = 1 + 1 = 2 \leq M \), we give node \( i_1 \) the label \( k + 1 = 2 \) and augment its weight to \( w_i + w_i'(k') = 2 \), since \( i_1 \) and \( i_2 \) are to be in the same cluster.
When node $i_2$ is encountered, we have $k' = 2$, $k = 1$, $w_{i_2}(k') = 3, w_{i_2}(k) = 1$. Table I for source trees again refers us to the conditions to which note 6 applies. Since $w_{i_2} + w_{i_2}(k') = 1 + 1 = 2 \leq M$, we give node $i_2$ label 3 and augment its weight to 2.

Finally, when the initial source node $a$ is encountered, we have $k' = 3$, $w_a + w_a(k') = 4 \geq M$, and node $a$ is assigned the label 4 according to substep c) of Step 4.

Fig. 5(d) shows the results of applying Step 5 to the relabeled tree. A successful reclustering has thus been obtained which has reduced delay to its minimum possible value.

V. LABELING PROCEDURE FOR ACYCLIC GRAPHS

A. Clustering with Replication of Nodes

The authors have been unsuccessful in developing an efficient algorithm for optimum clustering of networks which are not trees, except for certain special cases. However, it is possible to change the ground rules of the clustering problem so that labeling procedure of Section III (for rooted trees) can be meaningfully applied to such networks. This is accomplished by permitting "replication" of nodes of the network. In terms of an actual digital circuit, this means that extra copies of various logic gates may be created, and all such gates will be supplied with the same inputs, possibly necessitating an increase in the fan-out from other logic gates.

The complete clustering procedure is as follows.

**Step 0 and 1**: Label the nodes, as in the procedure for rooted trees described in Section III.

**Step 2**: Locate each node with the property that no successor of the node in question has the same label $k$ as the node itself. The node and each of its $k$-predecessors compose a cluster. Number the clusters $1, 2, \ldots, m$.

**Step 3**: Construct a new graph as follows:

a) For each node of the original graph, create a node $j_q$ for each cluster $q$ in which node $j$ is contained.

b) For each edge $(i, j)$ of the original graph, create an edge $(i_p, j_q)$ for each cluster $q$ in which node $j$ is contained, where

i) $p = q$ if $i$ and $j$ have the same label

ii) $p$ is chosen arbitrarily, otherwise.

The graph constructed in Step 3 contains as many copies of each node as there are distinct clusters in which it is contained.

The application of the labeling algorithm is illustrated in Fig. 6, with the restriction $M = 5$. It is clear that the original Rule 3 is violated, which required that each connected component of the subgraph of nodes given the same label was to contain $M$ or fewer nodes. For example, the subgraph containing nodes 1 through 12, excluding 8, with the label (0) clearly contains 11 nodes. However, we are able to transform the labeled graph into a clustered graph for which a) several nodes will be replicated,
b) the switching function realized by the gate network topologically equivalent to the graph is identical to the function realized by the initial network, and c) the delay is identical to the largest label appearing in the original graph. In Fig. 7 we display the resultant graph associated with the labeled graph of Fig. 6, wherein the nodes assigned to a given cluster are circled. It is seen that node 1 appears in 5 clusters, node 2 in 3 clusters, etc., thus indicating that 11 additional nodes are introduced by application of the algorithm.

It is of interest to estimate the amount of computation required for the execution of the labeling algorithm. Assume there are \( N \) nodes. For Steps 0 and 1, a computational growth rate of \( N^2 \) is implied. Likewise, Step 2 implies a growth rate of \( N^3 \), and Step 3, at worst, \( MN^2 \). Thus, the overall growth rate should be no worse than \( MN^3 \).

We assert that there exists no clustering technique, even if additional node replication is permitted, which will yield less delay than that obtained by the algorithm just described. We do not contend, however, that the amount of node replication is particularly economical; other procedures might yield the same delay with the creation of many fewer nodes. Below we shall describe methods for reducing the number of redundant nodes and the total number of clusters in the graphs obtained by this algorithm.

B. Reduction of Node Redundancy

It is clear, by referring to Fig. 7, that two or more clusters can sometimes be replaced by their union, without exceeding the constraint of \( M \) or fewer nodes per subgraph, thus enabling a reduction in the number of redundant nodes or in the total number of required clusters. For example, the distinct clusters, \{8, 14\}, \{8, 13, 20, 24\} can be replaced by the single cluster \{8, 13, 14, 20, 24\}, thus providing a graph for which node 8 is not required to be replicated and for which one less cluster is required.

Thus, in an attempt to reduce the total cost, let us consider the following two problems.

**Problem 1:** List all of the clusters resulting from the application of the procedure. Find a minimal reduced set of clusters, which results from forming the union of selected initial clusters, with the constraint that the cardinality of any cluster does not exceed \( M \).

It appears that the only technique for solving this problem is to formulate it as a covering problem and then to apply techniques developed by Lawler [3]. Such an approach is clearly not attractive for graphs which result in a large number of resultant clusters. An objective function which also reduces the overall cost, and which appears somewhat more tractable, is reflected in the following problem.

**Problem 2:** List all maximal sets of clusters which have the same label \( k \), \( k = 0, 1, \ldots \), and which contain nodes which collectively form a connected subgraph. For each such set of clusters, form the union of selected initial clusters such that the cardinality of any resultant cluster does not exceed \( M \), and such that the sum of the cardinalities of the clusters is minimal.

A heuristic solution to Problem 2 which may be optimal is presented in [4]. In any event, the graph of Fig. 7 can be transformed to the graph of Fig. 8, for which the number of redundant nodes is reduced from 11 to 5.

It is noted again that the clustering algorithm described in this section achieves minimal delay but not necessarily with minimum node or cluster redundancy. In Fig. 9 we display a clustering for the acyclic graph in question which achieves a delay of two without any node replication.

It is shown in [4] that an upper bound on the average node redundancy \( R(M, f) \) introduced is given by \( R(M, f) \leq M - M/f \), where \( f \) is once again the maximum out-degree of a node. However, on the basis of some experimental evidence, it appears that for most graphs, the degree of redundancy introduced by application of the algorithm is not significant.
Fig. 8. Optimal clustering containing fewer redundant nodes.

Fig. 9. A clustering technique requiring no redundancy.

VI. LABELING PROCEDURE FOR PIN LIMITATIONS AND OTHER CONSTRAINTS

A practical constraint encountered in the clustering of actual logic circuits is that of "pin" limitation. This refers to a restriction on the number of distinct signals (wires) that can be directed into and out of any given cluster. In the graphical representation of logic circuits, the number of signals directed into and out of a cluster is essentially equal to the number of distinct edges between nodes of the cluster and nodes of other clusters. (Of course, two or more edges may on occasion transmit the same signal, and can therefore share the same pin, but we shall ignore this possibility.)

Suitable modifications can be made in the algorithms of each of the previous sections in order to accommodate pin limitation. This is due to the fact that pin limitation is, at least for rooted and nonrooted trees, a "monotone" constraint, and any monotone constraint can be accommodated with little difficulty.

Definition: A constraint on the feasibility of clusters is said to be monotone if and only if any connected subset of nodes in a feasible cluster also compose a feasible cluster (cf. the definition in Lawler [3]).

For any graph for which clustering is to be performed, let us supply a "virtual" edge into each source node and a similar virtual edge out of each sink node. These will correspond to signals which are to be received from, or sent to, the world outside the network in question. Each of these signals requires a pin. For any given subset of nodes $S$, let $p(S)$ = the number of edges, including virtual
edges, which have exactly one end in the subset \( S \). Thus \( p(S) \) corresponds to the number of pins required for a cluster containing the subset \( S \). We leave it as an exercise for the reader to demonstrate that \( p(S) \leq P \), where \( P \) is a constant, is a monotone constraint for trees, but not for acyclic graphs generally.

Note that weight limitation of the type described in previous sections is itself a monotone constraint. Moreover, any combination of monotone constraints is itself equivalent to a single monotone constraint.

We are now in a position to generalize the previously described algorithms for a monotone constraint of the form \( p(S) \leq P \). In order to do so, we need a bit of notation. We shall let \( R(i, k) \) denote the set of \( k \)-predecessors of node \( i \), and \( S(i, k') \) the set of \( k' \)-successors external to the tree in question.

Consider the generalization of the labeling procedure for rooted trees.

**Step 0:** Label all source nodes 0.

**Step 1:** Find any unlabeled node \( i \), all of whose predecessors have been labeled. Let \( k \) be the largest label applied to any of these predecessor nodes. If \( p(R(i, k) \cup \{i\}) \leq P \), then apply label \( k \) to node \( i \); otherwise, apply label \( k+1 \).

Repeat Step 1 until all nodes are labeled.

Fig. 10 illustrates the application of this procedure for \( P = 4, M = 3 \).

In the case of nonrooted trees, Step 4 of the relabeling procedure is modified as follows.

**Step 4:** Assume the tree is a sink tree (modification of the procedure for source trees is obvious).

a) Label all source nodes 0.

b) Find any unlabeled node \( i \), all of whose predecessors have been relabeled. If this node is the sink node, pass to substep c). Let \( k \) be the largest label of any predecessor node, and \( k' \) the largest label of any successor node. Assign a new label to node \( i \), according to Table II. Repeat this step as often as necessary.

c) In the case of the root node, let \( k \) be the largest label of any predecessor node. If the value of this label is greater than \( D' - 1 \), go to Step 6.

Note that in Table II the instruction “replace \( \{i\} \) by \( \{i\} \cup S(i, k') \)” occurs in place of “add \( w_i(k') \) to \( w_i \).” The interpretation of this instruction should be fairly obvious; thereafter, all the nodes in the set \( \{i\} \cup S(i, k') \) are considered to be predecessors of any node of which \( i \) alone is a predecessor.

In the case of acyclic graphs, the labeling procedure is carried out exactly as it is for rooted trees, with the pin

---

**TABLE II**

**Summary of Relabeling Procedure for Pin Constraint for Sink Trees**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k + k' &gt; D' - 1 )</td>
<td>Go to Step 6</td>
</tr>
<tr>
<td>( k + k' = D' - 1 )</td>
<td>Assign node ( i ) the label ( k ) Replace ( {i} ) by ( {i} \cup S(i, k') ) Go to Step 6</td>
</tr>
<tr>
<td>( k + k' &lt; D' - 1 )</td>
<td>Assign node ( i ) the label ( k + 1 )</td>
</tr>
<tr>
<td>( p((i) \cup R(i, k)) &gt; P )</td>
<td>Assign node ( i ) the label ( k + 1 ) Replace ( {i} ) by ( {i} \cup S(i, k') ) Go to Step 6</td>
</tr>
<tr>
<td>( p((i) \cup R(i, k)) \leq P )</td>
<td>Assign node ( i ) the label ( k + 1 ) Replace ( {i} ) by ( {i} \cup S(i, k') ) Go to Step 6</td>
</tr>
</tbody>
</table>
count being determined, in the case of sink trees, by
the number of edges into the cluster, plus one (for the
edge out of the sink node of the cluster in question).

VII. CONCLUSIONS AND RECOMMENDATIONS FOR
FURTHER RESEARCH

In this paper we have effectively solved the problem
of assigning the gates of a combinational network to
larger modules, subject to practical constraints on the
number of gates and pins which a module can accommodate, and in such a way that the maximum network de-
lay is minimized.

We have considered the cases in which the network is
in the form of a 1) rooted tree, 2) a nonrooted tree, and
3) an acyclic graph. Computationally efficient al-
gorithms (in the sense of algebraic growth of computa-
tion with the number of nodes) have been developed for
the first two cases, and for the third case, when replica-
tion of the nodes is permitted. It appears that computer
programs could be written in which the algorithms
would require only simple operations with the adjacency
matrix of the graph.

The need for node replication for acyclic graphs
should not necessarily be viewed as a drawback of the
method, since in many cases the provision for replication reduces the delay compared with that required for
a nonredundant clustered network, and it is assumed
that minimization of delay is the goal we are seeking.
The algorithm admittedly does not yield minimum node
redundancy subject to minimum achievable delay, al-
though in most examples studied the redundancy is not
excessive.

We can cite the following two unsolved problems.

1) Minimization of delay for acyclic networks,
where no gate replication is permitted.

2) Attainment of minimum gate replication, sub-
ject to delay minimization, where gate replica-
tion is permitted.

In addition, we can cite the following additional ex-
tensions and generalizations.

3) Unequal delays on intermodule connections,
i.e., according to some arbitrary a priori assign-
ment.
4) Nonmonotone clustering constraints (see Sec-
 tion VI).
5) Alternative optimization criteria, e.g., minimi-
 zation of the number of modules, or the number
 of intermodule connections, subject to a bound
 on the maximum delay.
6) Evaluation of the clustering algorithms as part
 of a large-scale clustering-placement-routing
program.
7) Extension to sequential networks, i.e. networks
 containing directed cycles.

ACKNOWLEDGMENT

The authors wish to express their appreciation to J.
Goldberg of Stanford Research Institute for helpful dis-
cussions on many aspects of this paper.

REFERENCES

[1] M. A. Breuer, "General survey of design automation of digital
[3] E. L. Lawler, "Electrical assemblies with a minimum number of
interconnections," IEEE Trans. Electronic Computers (Corres-
limited networks," Stanford Research Institute, Menlo Park,
Calif., Final Rept., Project 5999, Contract AF 19(628)-5920
with Air Force Cambridge Research Labs, Bedford Mass.,
January 1968.