THE FOLDED TREE *

BY

ARTHUR W. BURKS,[1] ROBERT McNAUGHTON,[2] CARL H. POLLMAR,[1]
DON W. WARREN [1] AND JESSE B. WRIGHT [1]

Part II**

5. THE VALIDITY OF THE SPLITTING TECHNIQUE

In this section we show that, given any admissible n-term sequence $S'(1,1)$ containing no zeros, the procedure specified in Section 4 always results in a folded tree whose loading sequence $S(1,1) = S'(1,1)$. Thus we have a constructive proof that the admissibility of any sequence of positive integers is a sufficient condition for it to be the loading sequence of a folded tree. The most difficult part of the proof is to show that the splitting technique is admissibility-preserving, that is, to show that the sequences $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$ which result from $S'(i,j)$ by the splitting technique are admissible if $S'(i,j)$ is admissible.

(We do not give a complete proof here but rely on the results obtained in (4).[3] We shall, however, present enough of the proof to give an intuitive understanding of the argument.)

Theorem 10: If $S'(i,j)$ is an admissible sequence with at least two non-zero terms, then the two sequences which result from applying the splitting technique to it satisfy the unit condition.

Proof (here and in the following proofs we use the notation developed in Section 4): Since $S'(i,j)$ has at least two non-zero terms, d_1 must exist. $d_1 > 1$ since $M(S'(i,j))$ satisfies the partial sum condition. If $d_1 = 2$, then $b_1 = c_1 = 1$. If $d_1 > 2$, then $S'(i,j)$ cannot have exactly two non-zero terms, otherwise $S'(i,j)$ would not satisfy the total sum condition; hence, d_2 also exists and $c_1 = b_2 = 1$.

Theorem 11: Both of the sequences which result from applying the splitting technique to an admissible sequence satisfy the total sum condition.

---

* The writing of this paper and the research which it reports were done under the sponsorship of the Burroughs Corporation, Research Center, Paoli, Pa.

[1] The University of Michigan, Ann Arbor, Mich.

[2] Stanford University, Stanford, Calif.

** Part I appeared in this JOURNAL for July, 1955.

[3] The boldface numbers in parentheses refer to the references appended to Part I of this paper.

Proof: We need two lemmas.

Lemma 1. If $4 \leqq x \leqq p + 1$, then $\Delta_x = 0$ or 1; if $d_1 = 2$, then this is so for $x = 1,2,3$ as well.

The proof is by induction, and can be found on p. 33 of (4) (Lemma 1). (Note that "d_x" of this paper corresponds to "a_i" of (4).)

Lemma 2. $\Delta_{p+1} = 0$.

Proof: We show first that $\Delta_{p+1} = 0$ or 1. By Lemma 1 the only possible exceptions to this would be where $d_1 > 2$ and $p = 1$ or 2. If $p = 1$, then $d_1 = 2$, in order that $M(S'(i,j))$ satisfy the total sum condition. If $p = 2$, then $d_1 + d_2 = 6$ by the total sum condition. If $d_1 > 2$, then $d_1 = d_2 = 3$, since the d-sequence is monotonic non-decreasing. Here $c_1 = b_2 = 1$ and $b_1 = c_2 = 2$ so that $\Delta_3 = 0$. Having shown that in every case $\Delta_{p+1} = 0$ or 1, we must now show that the value is actually 0. As mentioned in the preceding section, for each x, $c_x + b_x = d_x$; hence,

$$\sum_{x=1}^{p} c_x + \sum_{x=1}^{p} b_x = \sum_{x=1}^{p} d_x.$$

But, since $M(S'(i,j))$ satisfies the total sum condition,

$$\sum_{x=1}^{p} d_x = 2^{p+1} - 2,$$

which is even. But Δ_{p+1} is precisely the difference between

$$\sum_{x=1}^{p} c_x \quad \text{and} \quad \sum_{x=1}^{p} b_x.$$

If Δ_{p+1} were 1, then their sum would have to be odd. Hence, $\Delta_{p+1} = 0$.

The theorem now follows directly. Since Lemma 2 entails that

$$\sum_{x=1}^{p} c_x \quad \text{and} \quad \sum_{x=1}^{p} b_x$$

are equal, it follows that

$$\sum_{x=1}^{p} d_x = 2 \sum_{x=1}^{p} b_x = 2 \sum_{x=1}^{p} c_x = 2^{p+1} - 2.$$

Hence,

$$\sum_{x=1}^{p} b_x = \sum_{x=1}^{p} c_x = 2^p - 1$$

and Theorem 11 is proved.

Theorem 12: If $b_1, \ldots, b_p$ and $c_1, \ldots, c_p$ are the sequences which result from the application of the splitting technique to an admissible

sequence $S'(i,j)$, then for every x such that $1 \leqq x \leqq p$,

$$\sum_{y=1}^{x} b_y \geqq 2^x - 1 \quad \text{and} \quad \sum_{y=1}^{x} c_y \geqq 2^x - 1.$$

This is proved as Lemma 3 on p. 34 of (4). (The "derived order," as the term is used in (4), is the order $b_1, b_2, \ldots, b_p$ and $c_1, c_2, \ldots, c_p$ as opposed to the monotonic order.) Roughly, Theorem 12 follows because $M(S'(i,j))$ satisfies the partial sum condition and the xth term of the b-sequence and the xth term of the c-sequence are usually obtained from the $(x + 1)$st term of $M(S'(i,j))$ in such a way that

$$\sum_{y=1}^{x} b_y \text{ is nearly equal to } \sum_{y=1}^{x} c_y.$$

Thus, $\sum_{y=1}^{x} b_y$ is approximately $\frac{1}{2} \sum_{y=1}^{x} d_y$.

Since $1, d_1, \ldots, d_x$ are the first $x + 1$ terms of $M(S'(i,j))$ and, since $S'(i,j)$ satisfies the partial sum condition,

$$1 + \sum_{y=1}^{x} d_y \geqq 2^{x+1} - 1$$

and hence

$$\tfrac{1}{2} \sum_{y=1}^{x} d_y \geqq 2^x - 1.$$

Theorem 13: If $S'(i,j)$ is admissible, then $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$ satisfy the partial sum condition.

This is proved as Theorem 3 on p. 40 of (4). In order to prove Theorem 13 one must extend the result of Theorem 12; for the b-sequence and c-sequence are not always in monotonic non-decreasing order, and a sequence satisfying the summation condition of Theorem 12 may no longer satisfy it when monotonized. Fortunately, it can be proved that after the third term the b-sequence and c-sequence are each, in the terminology of (4), quasi-monotonic, that is, for any x and x', if $4 \leqq x < x' \leqq p$, then $b_x \leqq b_{x'} + 1$ and $c_x \leqq c_{x'} + 1$. It is rather easy to show by virtue of this fact and Theorem 12 that, for each x,

$$\sum_{y=1}^{x} b'_y \geqq 2^x - 1,$$

where $b'_1, \ldots, b'_p$ is monotonic after the third term and results from $b_1, \ldots, b_p$ by interchanging the appropriate terms after b_3; similarly

for the c-sequence. (Cf. the theorem on p. 18 of (4).) The extension of this result to the case where the first three terms are rearranged to make the entire sequence monotonic is made in the proof in (4).

Theorem 14: For every $i < n$, if $S'(i,j)$ has n terms including exactly $i - 1$ zero terms and is admissible, and if $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$ are obtained from $S'(i,j)$ by the splitting technique, then $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$ are admissible and each contains exactly i zeros.

Proof: That these are admissible follows from Theorems 10, 11, and 13. $S'(i + 1,2j - 1)$ differs from $b_1, \ldots, b_p$ in at most the presence of zeros and the order of terms, which does not affect admissibility; similarly for $S'(i + 1,2j)$ and $c_1, \ldots, c_p$. To prove that they each have i zeros, it suffices to prove that there is one and only one unit term in $S'(i,j)$. That there is one follows from the fact that $S'(i,j)$ satisfies the unit condition. That there is only one follows from the fact that $S'(i,j)$ satisfies the partial sum condition; for if there were at least two, then the sum of the first two terms of $M(S'(i,j))$ would be $2 < 2^2 - 1$.

Theorem 15: Given $S'(1,1)$ as an admissible n-term sequence without zeros, the procedure specified in Section 4 results in an n-bay folded tree whose $S(1,1) = S'(1,1)$.

Proof: From Theorem 14 it follows that for any $j < 2^{n-1}$ the sequence $S'(n,j)$ has $n - 1$ zeros and is admissible. Its one non-zero term, say, a_k, must be unity by the total sum condition, and so $V(n,j)$ is labeled P_k. Therefore, (1) $S'(n,j)$ is $S(n,j)$ and $T(n,j)$ is a folded tree.

We now go on to prove that (2) for any i and j, if $S'(i + 1,2j - 1)$ is $S(i + 1,2j - 1)$ and $S'(i + 1,2j)$ is $S(i + 1,2j)$, and if $T(i + 1,2j - 1)$ and $T(i + 1,2j)$ are folded trees, then $S'(i,j)$ is $S(i,j)$ and $T(i,j)$ is a folded tree. There is a set

$$\{P_{m_1}, \ldots, P_{m_{n-i}}\}$$

of $n - i$ distinct labels such that each chain of $T(i + 1,2j - 1)$ contains exactly one vertex labeled with each member of the set. The m_1th, $\ldots$, m_{n-i}th terms of $S(i + 1,2j - 1)$, which is $S'(i + 1,2j - 1)$, are exactly those terms which are non-zero terms. There is a similar set for $T(i + 1,2j)$. Now this set is identical to the set for $T(i + 1, 2j - 1)$, since for any q the qth term of $S'(i + 1,2j - 1)$ will be made zero by the splitting technique if and only if the qth term of $S'(i + 1,2j)$ is made zero. If $V(i,j)$ is labeled P_k, then k is not one of the $m_1, \ldots, m_{n-i}$; this is so because the kth term of $S'(i,j)$ is one and, therefore,

the kth term of $S'(i + 1,2j - 1)$ and the kth term of $S'(i + 1,2j)$ are each zero. From this it follows that every chain of $T(i,j)$ has exactly one vertex labeled with each member of the set

$$\{P_k, P_{m_1}, \ldots, P_{m_i}\},$$

since a chain of $T(i,j)$ is either a chain of $T(i + 1,2j - 1)$ or a chain of $T(i + 1,2j)$ with $V(i,j)$ and its vertex-input added. Hence $T(i,j)$ is a folded tree. Now each term of $S'(i,j)$, except $a'_k(i,j) = 1$, is equal to the sum of the corresponding terms of $S'(i + 1,2j - 1)$ and $S'(i + 1, 2j)$; $a'_k(i + 1,2j - 1) = a'_k(i + 1,2j) = 0$. (This follows from the specification of the splitting technique.) Since the vertices of $T(i,j)$ are those of $T(i + 1,2j - 1)$ and $T(i + 1,2j)$ together with $V(i,j)$, it follows, from the fact that $S'(i + 1,2j - 1)$ is $S(i + 1,2j - 1)$ and $S'(i + 1,2j)$ is $S(i + 1,2j)$, that $S'(i,j)$ is $S(i,j)$.

From (1) and (2) it follows immediately by induction that $S'(1,1)$ is $S(1,1)$ and $T(1,1)$ is a folded tree.

6. THE ECONOMY OF THE FOLDED TREE

The question arises as to whether circuits represented by folded trees are the most "economical" ones which can function as complete decoding circuits. As we have indicated earlier (Section 2) the answer is in the negative so far as electronic digital computing circuits are concerned (see also Section 5 of (2)). However, folded tree relay transfer contact nets are probably the most economical (in a sense to be defined) of all complete decoding relay transfer contact nets. To formulate this proposition precisely we must delimit the class of diagrams whose realizations are all such complete decoding relay transfer contact nets.

Our definition of vertex diagram at the beginning of Section 2 was motivated by two considerations: (1) that a relay transfer contact is well represented by a vertex with a single vertex-input and two vertex-outputs, and (2) that in a relay transfer contact net the relay transfer contacts can be arbitrarily connected. Hence any relay transfer contact net can be represented by some vertex diagram. In this section we go on to define an n-label complete decoding vertex diagram in such a way that every transfer contact net which performs a complete decoding function is represented by a diagram of this kind. Our belief that folded tree relay transfer contact nets are probably the most economical of all complete decoding relay transfer contact nets can now be more precisely stated as the conjecture: The n-bay folded tree has the minimal number of vertices of any n-label complete decoding vertex diagram.

The objection may be made that the minimality of the number of vertices of a complete decoding vertex diagram is not a sufficient condition for its realization by relays to be minimal in cost, for the cost of a relay transfer contact net depends not only on the number of

transfer contacts in it but also on the number of relay coils required to operate it. To particularize this objection, consider the problem of constructing a complete decoding net using only relays with eight transfer contacts each; here the number of relays required is a better indication of the cost of the net than the number of transfer contacts it contains.

There is a certain force to this objection. However, it is to a large extent mitigated by our previous folding results. For if an n-bay standard tree circuit has a minimal number of contacts, we can in practice use that folded tree circuit which of all n-bay folded trees has the least number of relay coils. For an example see the last part of Section 2. When relays with different numbers of contacts are available at costs which are not directly proportional to the number of contacts they contain, then a different folding can be employed to minimize the total cost of the circuit.

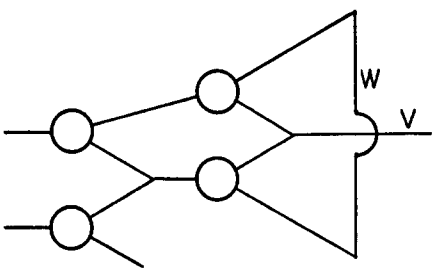

FIG. 9.

We have not proved our minimality conjecture, but in this section we present a partial result in that direction. To this end we will introduce a certain subclass of the class of n-label complete decoding vertex diagrams, namely, the subclass of all n-label progressive diagrams. We shall prove that the n-bay folded tree has the minimal number of vertices of an n-label progressive diagram. It seems, intuitively, that an n-label complete decoding vertex diagram which is not an n-label progressive diagram should have at least as many vertices as an n-bay folded tree, and it is on this ground that we make our conjecture.

We now proceed to carry out the program sketched above. Since all the vertex diagrams considered in previous sections were trees, we first present an example of a vertex diagram that is not a tree (Fig. 9). (Note the use of the loop in the wire W of Fig. 9 to indicate that the wire W does not touch the wire V.)

Some of the concepts already introduced in connection with trees must be generalized to apply to arbitrary vertex diagrams. First, we require a more general method of describing the way states of wires are determined. To accomplish this we define the notion of the connection of two wires.

We say that two wires are *directly connected* when they touch each other. A vertex is *directly connected* to a wire whenever the wire is either its vertex-input, or its upper (lower) vertex-output when the vertex is in the upper (lower) setting. Two wires W and W' are *connected* if there is a sequence of wires and vertices $X_1, \ldots, X_n (n \geqq 1)$ such that W is X_1, W' is X_n, and such that, for each $i < n$, X_i is directly connected to $X_i + 1$. By taking $n = 1$ we see that any wire is always connected to itself. The sequence $X_1, \ldots, X_n$ is a *connection* of W and W'. Thus, whether two wires in a diagram are connected usually depends upon the settings of some of the vertices of the diagram.

An *n-label complete decoding vertex diagram* (with designated diagram input and diagram outputs) is a vertex diagram in which the following hold.

1. Each vertex V has exactly one label from a set of n distinct labels.
2. For each label there is at least one vertex with that label.
3. There is exactly one wire K designated as the *diagram input.*
4. A *diagram state* is a definite assignment of the vertex settings such that (*a*) all the vertices with the same label are set the same, and (*b*) a wire is in state 1 if and only if it is connected to the diagram input.
5. For each diagram state, there is at least one wire which is in state 1 in that diagram state and in state 0 in every other diagram state, and for each diagram state one such wire is designated (arbitrarily, if there is more than one) as a *diagram output* of the diagram.

The diagram input is in state 1 in any diagram state since it is always connected to itself. Since there are n labels for the vertices, there are 2^n diagram states, and, therefore, 2^n diagram outputs. For any diagram output Q, let $S(Q)$ be the diagram state in which Q is in state 1.

Sometimes in these diagrams a vertex may have the state of its vertex-input depend on the state of one of its vertex-outputs. For example, consider the uppermost vertex of Fig. 10. When it is in its lower setting, the state of its vertex input Q_2 depends upon the state of its lower vertex output. For this reason the terms "vertex-input" and "vertex-output" are not as appropriate for the more general class of vertex diagrams as they are for the class of trees.

Obviously, trees are vertex diagrams, and by Theorem 1 n-bay folded trees are complete decoding n-label vertex diagrams.

For an arbitrary connection $X_1, \ldots, X_p$, if X_k is directly connected to X_m for $m > k + 1$, then $X_{k+1}, \ldots, X_{m-1}$ are *superfluous;* the sequence with them deleted is still a connection. It is easy to see, therefore, that in a diagram state, S, if there is a connection between W and W', then there is a connection between them in S without any

superfluous vertices or wires. A *chain* of a diagram output Q is a sequence $X_1, \ldots, X_p$ of wires and vertices which in $S(Q)$ is a connection of the input K and Q without any superfluous elements. Obviously, the chain as defined in Section 2 is a chain in this sense.

If $X_1, \ldots, X_m$ is a chain C (where X_1 is the diagram input K and X_m is a diagram output), then for any i,j such that $i < j$ we say that X_i is *earlier* than X_j in C. If X_i is a vertex, then X_{i-1} and X_{i+1} are wires of the vertex, one being the vertex-input of X_i and the other being one of the vertex-outputs of X_i; we say that X_{i-1} is the *early wire* and X_{i+1} the *late wire* of the vertex X_i in the chain C. If the early wire of a vertex in a chain is the vertex-input, then we say the vertex is oriented *forward* in the chain; if the early wire is a vertex-output, then the vertex is oriented *backward*. For example, the vertices labeled P_2 in the chains of Q_1 and Q_2 in Fig. 10 are backward, whereas the vertex

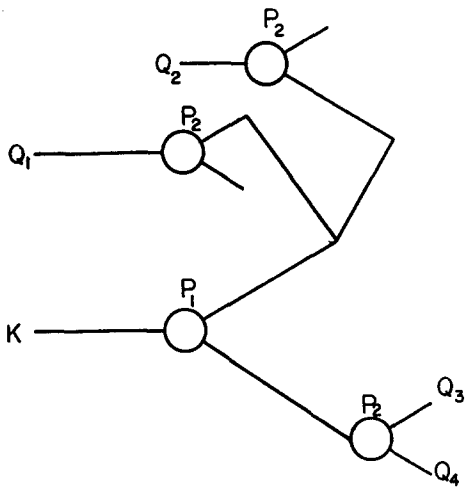

FIG. 10.

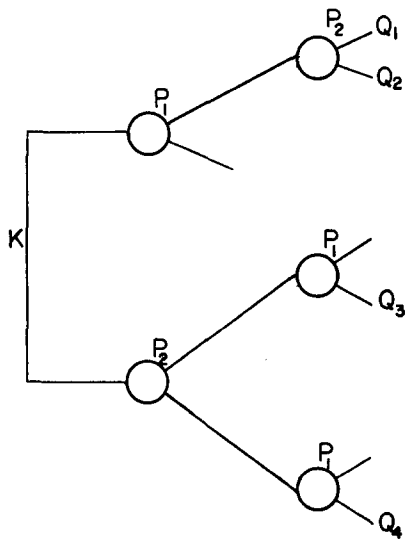

FIG. 11.

labeled P_2 in the chains of Q_3 and Q_4 is forward in both those chains. The vertex labeled P_1 is forward in all chains.

A *progressive diagram* is a complete decoding vertex diagram in which each output Q has at least one chain in which all vertices are forward. The folded tree is a progressive diagram while Fig. 10 is a complete decoding vertex diagram which is not progressive.

An output Q of a progressive diagram may have more than one chain in which all vertices are forward. It is convenient to pick out for each output Q of a progressive diagram one such chain and refer to it as $C(Q)$, or the *selected chain* of the output.

Where Q and Q' are two distinct diagram outputs of a complete decoding vertex diagram, we define $V(Q,Q')$ to be the latest vertex V

in $C(Q)$ whose early wire is in state 1 in $S(Q')$ and whose late wire is in state 0 in $S(Q')$. Such a vertex must exist since the diagram input K is in state 1 in $S(Q')$ but Q is in state 0 in $S(Q')$; the latest wire of $C(Q)$ which is in state 1 in $S(Q')$ must be the early wire of a vertex in $C(Q)$. For example, in Fig. 11 $V(Q_1,Q_4)$ is the vertex labeled P_1 in $C(Q_1)$ and $V(Q_4,Q_1)$ is the vertex labeled P_2 in $C(Q_4)$. This example shows, incidentally, that $V(Q,Q')$ and $V(Q',Q)$ are not always identical.

Whenever A is any set of outputs of a progressive diagram we let $D(A)$ be the set of just those vertices $V(Q,Q')$ where Q and Q' are both in A. If A has only one output Q, then $D(A)$ is the null set, since there is no $V(Q,Q)$. For any set A and any output Q not in A, $A + \{Q\}$ is the set whose members are Q and all the members of A.

Theorem 16: If A is a set of one or more outputs of a progressive diagram, and if Q is an output not in A, then there is a vertex in $D(A + \{Q\})$ which is not in $D(A)$.

Proof: Let V be the latest vertex $V(Q,Q')$ in $C(Q)$ where Q' is in A. We shall prove that V is not in $D(A)$, from which Theorem 16 follows directly since V must be in $D(A + \{Q\})$.

We shall use the *reductio ad absurdum* method. Suppose that V is in $D(A)$. Then there are diagram outputs Q'' and Q''' in A such that V is $V(Q'',Q''')$. Let x be the early wire of V in $C(Q)$ and y the late wire. Since the diagram is progressive and since V is in $C(Q)$, x is the vertex-input and y is a vertex-output of V. Let z be the other vertex-output of V. Since V is $V(Q'',Q''')$, x is in $C(Q'')$ and either y or z is in $C(Q'')$.

Case I. y is in $C(Q'')$. Then y is in state 1 in $S(Q'')$. The latest wire in $C(Q)$ which is in state 1 in $S(Q'')$, then, can be no earlier than y, and hence $V(Q,Q'')$ is later than V in $C(Q)$, contrary to the original stipulation that V be the latest such vertex in $C(Q)$.

Case II. z is in $C(Q'')$. Since V is $V(Q'',Q''')$, x must be in state 1 in $S(Q''')$, and z in state 0. This means that the setting of the vertex V in $S(Q''')$ must be such that y is in state 1 in $S(Q''')$. Reasoning as in Case I, then, we can show that $V(Q,Q''')$ is later than V in $C(Q)$, which is likewise contradictory. This completes the proof of Theorem 16.

Theorem 17: In a progressive diagram, if A is a set of k outputs then there are at least $k - 1$ vertices in $D(A)$.

Proof: Let $A_2, \ldots, A_k$ be a sequence of subsets of A such that A_k is A, A_i has exactly i members, and, if $2 \leqq i \leqq k - 1$, A_i is a proper subset of A_{i+1}. Thus, A_{i+1} has just one member besides those of A_i. There is at least one vertex in $D(A_2)$; and by Theorem 16 $D(A_{i+1})$ has

at least one more vertex than $D(A_i)$. Hence, $D(A_k)$ has at least $k - 1$ vertices.

Theorem 18: In the class of n-label progressive diagrams the n-bay folded tree has a minimal number of vertices.

Proof: In an n-label complete decoding vertex diagram there are 2^n outputs. If A is the set of all these outputs, then $D(A)$ must have at least $2^n - 1$ vertices. Hence, the diagram must have at least $2^n - 1$ vertices which is the number of vertices in the n-bay folded tree.

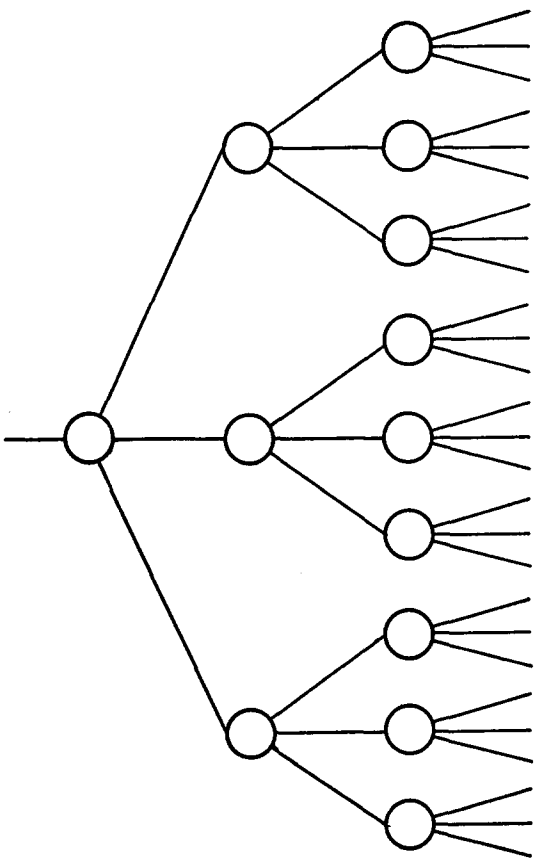

FIG. 12.

7. A GENERALIZATION OF THE FOLDED TREE

In this section we shall consider generalized folded trees containing vertices, all of which have the same arbitrary number of vertex outputs and possible settings. A vertex with m vertex-outputs is called an m-order vertex and its m settings are the first setting, the second setting, . . . , the mth setting. The ith *vertex output* is the ith right-hand wire from the top. (See Fig. 12.)

A generalized n-bay folded tree containing vertices of order m,

called an n-bay *m-order folded tree*, obviously contains

$$\sum_{x=1}^{n} m^{x-1} = (m^n - 1)/(m - 1)$$

vertices. Thus an n-bay folded tree as previously defined has order 2. The definition of "complete decoding" is the same as that given in Section 2, and the generalization of Theorem 1 goes through quite easily.

It is not difficult to see what sort of physical realization an m-order vertex can have, either in relay circuits or in electronic digital computing circuits. In relay circuits the m-order vertex represents a single-pole m-throw switch, for example, a stepping switch. In electronic digital computing circuits the m-order vertex can represent an arrangement of m conjunction elements generalized from the arrangement of Fig. 2.

As in Section 2 we ask the question, for a given loading sequence, $a_1, \ldots, a_n$, is there an n-bay m-order folded tree having a_1 vertices labeled $P_1, \ldots,$ and a_n vertices labeled P_n? We have a generalized condition of "admissibility" which we can prove to be necessary and which we conjecture to be sufficient if the sequence contains no zeros. Our generalized condition of admissibility involves four conditions, as compared with only three in Section 3.

A sequence satisfies the *unit condition* if there is a 1 somewhere in the sequence. It satisfies the *total sum condition* if the sum of all the terms is equal to

$$(m^p - 1)/(m - 1) = \sum_{x=1}^{p} m^{x-1}$$

where p is the number of non-zero terms. A sequence S satisfies the *partial sum condition* if, for each $k \leqq p$, the sum of the first k terms of $M(S)$ is greater than or equal to

$$(m^k - 1)/(m - 1) = \sum_{x=1}^{k} m^{x-1}.$$

It satisfies the *congruence condition* if, for each term a_k, $a_k \equiv 1 \pmod{m - 1}$, that is, $a_k - 1$ is divisible by $m - 1$. A sequence is *admissible* if it satisfies all four conditions. Note that admissibility as defined in Section 3 is a special case ($m = 2$) of this more general notion of admissibility, for any sequence of integers satisfies the congruence condition when $m = 2$.

Theorem 19: The loading sequence $S(1,1)$ of an n-bay m-order folded tree is an admissible sequence of n non-zero terms.

Proof: Obviously, $S(1,1)$ must satisfy the unit condition and the total sum condition. The reader can reread Theorem 8 and its proof and see that it very easily generalizes from the case ($m = 2$) to prove that $S(1,1)$ satisfies the partial sum condition for any m. To demon-

strate that $S(1,1)$ satisfies the congruence condition we consider any n-bay m-order folded tree having $a_k(1,1)$ vertices labeled P_k. Suppose that for each $h \leqq n$ there are k_h vertices labeled P_k in the hth bay. There are m^n chains and each chain must have exactly one vertex labeled P_k.

But a vertex in the hth bay is on exactly m^{n-h+1} chains. Hence (1) the number of chains is equal to

$$\sum_{h=1}^{n} k_h m^{n-h+1} = m^n.$$

By elementary number theory we know that for any non-negative integer x, $m^x \equiv 1 (\text{mod } m-1)$. Thus, for each h, $k_h m^{n-h+1} \equiv k_h (\text{mod } m-1)$. Hence (2)

$$\sum_{h=1}^{n} k_h m^{n-h+1} \equiv \sum_{h=1}^{n} k_h (\text{mod } m-1).$$

From (1) and (2), since $m^n \equiv 1 (\text{mod } m-1)$, it follows that

$$a_k(1,1) = \sum_{h=1}^{n} k_h \equiv 1 (\text{mod } m-1).$$

This completes our proof of Theorem 19.

APPENDIX

Interchange and the Folded Tree

The phrase "folded tree" is appropriate because a folded tree can be obtained from the standard tree of Section 2 by the technique of "folding." That technique, perhaps better described as "interchange," can be defined as follows. An n-bay labeled-tree $T'(1,1)$ results from an n-bay labeled-tree $T(1,1)$ by an *interchange* of P_h and P_k in the minor tree $T(i,j)$ when all the vertices labeled P_h in $T(i,j)$ are labeled P_k in $T'(i,j)$ and vice versa, all other vertices of $T'(1,1)$ being labeled the same as in $T(1,1)$.

By referring to the definition of "folded tree," it is not difficult to see (1) that if $T'(1,1)$ is obtained from $T(1,1)$ by interchange, then $T'(1,1)$ is a folded tree if and only if $T(1,1)$ is.

It can also be shown (2) that for any n-bay folded trees $T(1,1)$ and $T'(1,1)$ having the same set of labels, $T'(1,1)$ can be obtained from $T(1,1)$ by a sequence of interchanges. For suppose $T_1(1,1)$, $T_2(1,1)$, . . . , is a sequence of distinct n-bay labeled-trees where $T_1(1,1)$ is $T(1,1)$ and where $T_{x+1}(1,1)$ is obtained from $T_x(1,1)$ by the following process. Having ordered all the vertices of $T_x(1,1)$ in the sequence $V_x(1,1)$, $V_x(2,1)$, $V_x(2,2)$, $V_x(3,1)$, $V_x(3,2)$, . . . , we consider the first vertex $V_x(i,j)$ which has a label different from the corresponding vertex $V'(i,j)$ of $T'(1,1)$. Suppose that P_h and P_k are the labels of $V_x(i,j)$ and $V'(i,j)$, respectively. Then $T_{x+1}(1,1)$ results from $T_x(1,1)$ by interchange of P_h and P_k in $T_x(i,j)$. It is easily seen that P_k must label at least two vertices in $T_x(i,j)$, so the interchange can always be made if $T_x(1,1)$ is not identical with $T'(1,1)$. Obviously, then the label of $V_{x+1}(i,j)$ and the labels of all vertices of $T_{x+1}(1,1)$ which precede $V_{x+1}(i,j)$ in the ordering of vertices mentioned above are the same as the labels of the corresponding vertices of $T'(1,1)$. It is not difficult to see that the sequence $T_1(1,1)$, $T_2(1,1)$, . . . has a last member $T_q(1,1)$ which must be $T'(1,1)$.

Since a standard tree is a folded tree, it follows from (1) and (2) that a necessary and sufficient condition that an n-bay labeled-tree with labels P_1, . . . , P_n be a folded tree is that it be obtainable from the n-bay standard tree by a sequence of interchanges.

Everything asserted in this appendix is true also of trees all of whose vertices are of order $m > 2$.