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 I **

1. INTRODUCTION

The problem of constructing circuits which perform a certain function and have some formal properties contributing to engineering efficiency was considered by Shannon (1).[3] Part of Shannon's problem (Part II of (1), pp. 81–89) was to formulate an arithmetic condition such that for any sequence of positive integers satisfying that condition a tree circuit can be constructed whose load distribution is given by that sequence. That condition is identical with our notion of "admissibility," defined in Section 3 of this paper.

The present paper takes up this same problem, but is entirely self-contained. No reference to Shannon's paper is necessary either for definitions of concepts or for proofs of theorems. Our results go beyond those of Shannon, both in that we prove the necessity of the admissibility condition and also in that we give a constructive technique, helpful to a practical engineer, for constructing a folded tree with any given load distribution satisfying the condition of admissibility. To obtain these results we must formulate a precise definition of the term "folded tree." Although Shannon does not give a precise definition of any corresponding concept, it is clear that our precisely defined concept applies to the circuits which he considers.

We shall use some of the concepts of our previous paper (2), but we define them anew here, so no acquaintance with that paper is presupposed. Our method, here as in (2), is to discuss diagrams rather than circuits directly. The diagrams may be realized by circuits of various different kinds, for example, relay transfer contact nets discussed in (1), and electronic digital computing circuits. The advantages of this approach are (1) that the range of application of results is wider, and (2) that the problems, being abstract, can be solved within pure mathematics. Our methods and results are intimately connected with

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

The authors wish to thank Dr. Irving Copi for many helpful suggestions and also Dr. Frank Harrary for his suggestions.

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

[2] Stanford University, Stanford, Calif.

** Part II will be published in this JOURNAL for August, 1955.

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

the field of mathematics known as the theory of linear graphs; but they are not an application of any previously known materials of that field, and consequently we presuppose no knowledge of it. Although our definitions and theorems concern diagrams, we shall frequently comment on their significance for physical circuits.

2. DEFINITION OF FOLDED TREE

In this section we formulate a precise definition of the term "folded tree" and show that the diagrams covered by this term are suitable for representing some familiar circuits constructed out of standard devices.

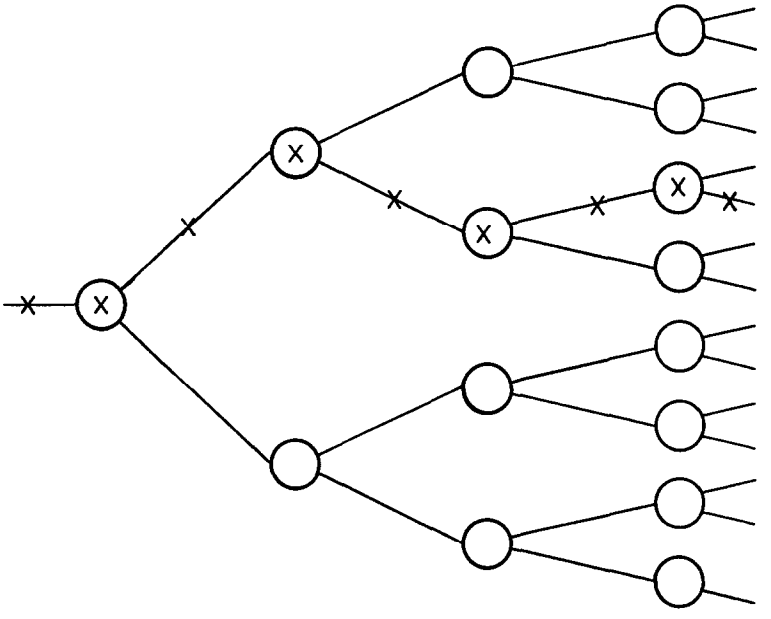

FIG. 1.

We shall be concerned with *vertex diagrams* which are arrangements of small circles and straight lines, such that (1) each circle has just three lines touching its circumference, one on its left and two on its right, (2) each end of each line may touch either a circle (as in Fig. 1) or the ends of any number of other lines (as in Fig. 10) or it may touch nothing (as in Fig. 1), and (3) no circles touch each other. (In this paper if a word is italicized in a sentence, then the word is defined in that sentence.) The circles are called *vertices* and the lines *wires*; the left-hand wire is called the *vertex-input*, the upper right-hand wire is called the *upper vertex-output*, and the lower right-hand wire, the *lower vertex-output*. The usage here of "output" and "input" is nearer to that of (1) than to (2).

The first vertex diagram we introduce is an *n-bay tree*, for which the following recursive definition is provided:

1. A *1-bay tree* consists of just one vertex with its input and output wires; this vertex is the *first* (and only) *bay* of the 1-bay tree;

2. An $(i + 1)$-*bay tree* results from an i-bay tree when, to each vertex-output u of a vertex in the ith bay, a new vertex is joined so that u is the vertex-input of the new vertex; the new vertices constitute the $(i + 1)$th bay;

3. An n-bay tree has only the vertices and wires provided for in 1 and 2; no diagrams other than those so provided for are trees.

The *input* of an n-bay tree is the vertex-input of the vertex of the first bay. An *output* of an n-bay tree is a vertex-output of a vertex in the nth bay. Note that the tree input is the only wire in the tree which is not a vertex-output, and that the tree outputs are the only wires which are not vertex-inputs. Note also that there are 2^n outputs of an n-bay tree. The result of deleting the crosses (but no lines or circles) from Fig. 1 is a 4-bay tree. (Note that each bay in this figure is simply a vertical column of vertices. This will be true of all figures of trees in this paper, although the definition of "tree" does not require this.)

A wire is to be understood as being at any one moment in either of two *states*, state 1 or state 0, and a vertex as being in just one of two *settings*, an upper setting or a lower setting. The state of a vertex-output is determined by the setting of the vertex and the state of the vertex-input: if the vertex-input is in state 1 and if the vertex is in the upper setting, then the upper vertex-output is in state 1 and the lower vertex-output in state 0; if the vertex-input is in state 1 and the vertex in the lower setting, then the upper vertex-output is in state 0 and the lower vertex-output in state 1; if the vertex-input is in state 0, then regardless of the setting of the vertex the state of both vertex-outputs is 0.

Before proceeding we will relate trees to the net diagrams of (2) and described two physical realizations of trees. In the net diagram of Fig. 2 each square is a conjunction ("and") element whose output is in state 1 if and only if both its inputs are in state 1. A vertex, together with its vertex-input and vertex-outputs, and the net of Fig. 2 represent the same type of circuit if exactly one of the input pair b,b' is in state 1 at any one time. (The use of the word "input" in this connection is similar to the use in (2); it differs significantly from the use of the word in the major part of this paper.) Wire a of Fig. 2 is the vertex-input, and wires c and d are the vertex-outputs, and the vertex is in the upper setting just in case b is in state 1. Nets like the one of Fig. 2 may clearly be combined to form trees.

A conjunction element may be realized by a crystal diode or vacuum tube circuit whose output voltage is high (state 1) whenever both circuit inputs are at high voltage, otherwise the circuit output voltage is low (state 0). Two such circuits, interconnected as in Fig. 2, constitute a physical interpretation of a vertex with its vertex input and two vertex-outputs. Wire c is in state 1 whenever a and b are both in state 1, and d is in state 1 whenever a and b' are both in state 1.

A vertex may also be realized by a relay transfer element. A relay transfer element is a (mechanical) single-pole double-throw switch with the wire of the pole realizing the vertex-input and the wires from the output contacts realizing the vertex-outputs. The transfer element

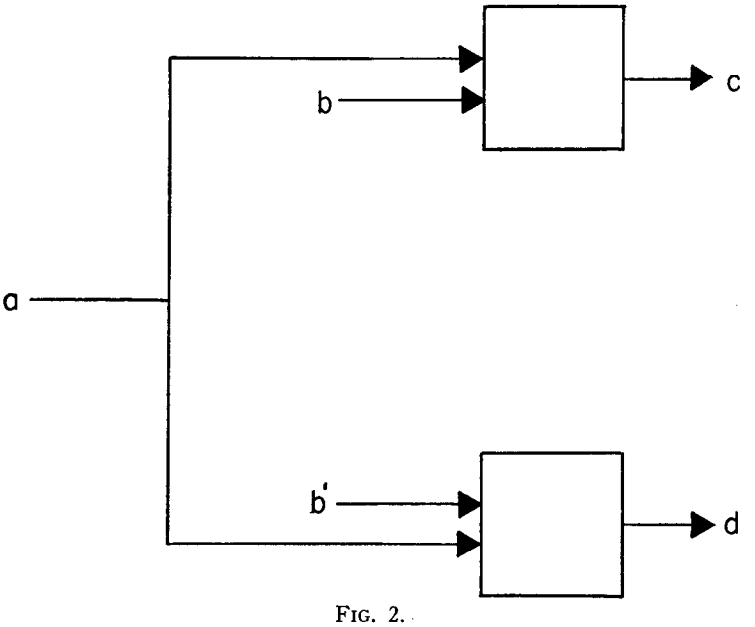

FIG. 2.

is controlled by a magnet and spring working in opposition, the two positions being represented by the two vertex settings. Thus, if current is flowing (state 1) in the vertex-input, it flows in the upper vertex-output whenever the relay is not activated and in the lower vertex-output whenever the relay is activated.

It should be noted that our vertex diagrams (trees and the diagrams of Section 6) are particularly suited to represent relay contact nets composed of transfer elements, namely, single-pole double-throw switches, electro-magnetically controlled. In such electrical nets the wires to the device (magnet) which controls the settings of the transfer contacts are not connected into the contact net itself. That our vertex diagrams do not show the wires which determine the vertex settings is intended to represent that feature. In contrast the wires b,b' of an electronic circuit of the sort represented by Fig. 2 are, in

general, like a, c, and d, connected to other components of the circuit. Electronic circuits can thereby realize nets which are more complex than vertex diagrams. Hence, the class of circuits represented by vertex diagrams (in the broad sense of Section 6) is narrower than the class of circuits represented by diagrams constructed from conjunction elements. It turns out, for example, that in a certain sense of "cost" the tree is probably a minimal diagram in the first class, while it is not at all minimal in the second class (see Section 6). To put the point alternatively, a generally efficient way to do complete decoding with relays is by means of a tree, but that is not generally an efficient way to do complete decoding with vacuum tubes or crystal diodes.

A *chain* of an n-bay tree is a sequence $X_1, \ldots, X_{2n+1}$, where X_1 is the tree input and where, for $i \leqq 2n$, if X_i is a vertex-input of some vertex, then X_{i+1} is that vertex and, if X_i is a vertex, then X_{i+1} is one of its vertex-outputs. It follows that X_{2n+1} is a tree output. In

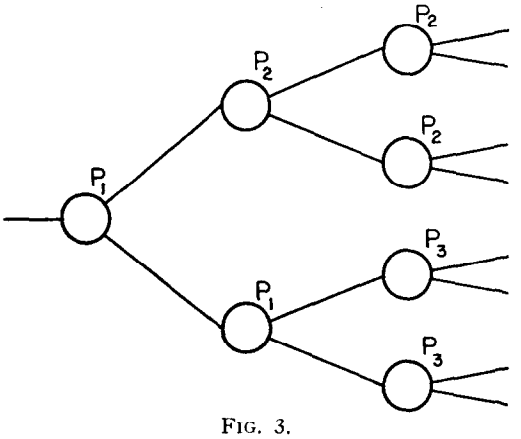

FIG. 3.

Fig. 1 the vertices and wires marked with crosses constitute a chain. It is obvious that each tree output is a member of one and only one chain.

In an n-bay tree the settings of the vertices and the state of the tree input vary independently of each other, but once these are determined the state of each wire is determined. If the tree input is in state 0, then every wire is in state 0, regardless of the settings of the vertices. If the tree input is in state 1, then there will be a chain whose every wire is in state 1 and such that every wire not on the chain is in state 0. What chain it is will depend on the settings of the vertices, since the state of each vertex-output is determined by the setting of the vertex and the state of its vertex-input.

We are interested in trees in which the settings of the vertices do not vary completely independently of each other, but whose vertices are partitioned into a number of classes, all the vertices of each class being in the same setting at any one time. In such trees we indicate the

class to which each vertex belongs by affixing the same label to every vertex of a given class, vertices which belong to different classes being given different labels. We call such a tree a "labeled-tree," and define it as follows. An *n-bay labeled-tree* is formed from an n-bay tree by marking each vertex with exactly one label from a set of m labels, and using each of the m labels to mark at least one vertex; m can be less than, equal to, or greater than n. We require, of course, that all vertices having the same label be in the same setting. Figure 3 is a 3-bay labeled-tree in which $m = n$.

An *n-bay folded tree* is an n-bay labeled-tree in which there are exactly n labels (that is, $m = n$) and, for every chain and for each label, there is at least one vertex with that label in the chain. It follows that there is exactly one vertex in each chain with a given label, since there

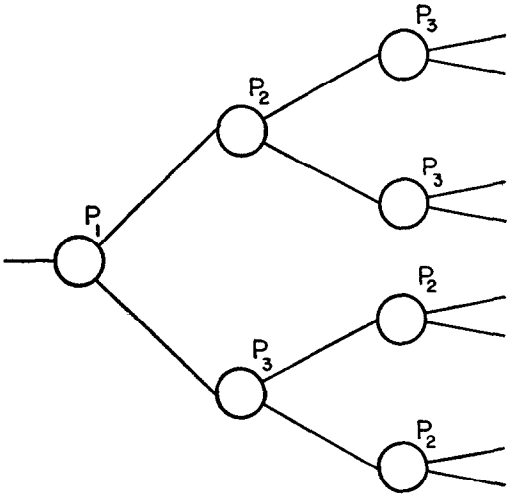

FIG. 4.

are n labels and n vertices in a chain. A remark on the appropriateness of the term "folded" in this connection will be made in the Appendix. Note that Fig. 4 is a folded tree, but Fig. 3 is not.

Folded trees are important among labeled-trees because they can function as "complete decoding nets" in the sense given in (2). We must explain what this means and why it is so. A circuit represented by a labeled-tree performs its function when the tree input is in state 1 (for example, after all the transfer contacts of a relay contact net are set, an electrical signal introduced at the tree input will emerge at the desired output). We define a *tree state* of a labeled-tree to be a definite assignment of its vertex settings such that all vertices with the same label are set the same, and such that the tree input is in state 1. It is clear that there are 2^m different tree states for a labeled-tree having m distinct labels.

We say that an n-bay labeled-tree is *complete decoding* if and only if the number of labels m is equal to n, and for each tree state there is at least one tree output which is in state 1 in that tree state and state 0 in every other tree state. It follows that there is only one tree output in state 1 in any tree state of a complete decoding n-bay labeled-tree, since it has exactly 2^n tree states and 2^n tree outputs. (A nearly identical concept of complete decoding is discussed in Section 3 of (2).)

Theorem 1: A necessary and sufficient condition for a labeled-tree to be complete decoding is that it be a folded tree.

Proof of necessity: Consider an n-bay labeled-tree which is complete decoding. Consider any tree state S and the tree output Q which is in state 1 only in S. There is only one chain C which includes Q. Suppose that, for some label P_i, there is no vertex in C labeled P_i. Now let S' be a different tree state which coincides with S except for the settings of the vertices labeled P_i. Q would be in state 1 in S' as well as in S, contrary to the assumption that the tree is complete decoding. Hence, every tree state determines one chain such that for each label there is at least one vertex with that label in the chain. Different tree states determine different chains; and since there are 2^n different tree states, there must be 2^n different chains, each of which has the property that for each label there is at least one vertex with that label in the chain. But these are all the chains that there are, since an n-bay labeled-tree contains exactly 2^n chains. Hence, any complete decoding labeled-tree is a folded tree.

Proof of sufficiency: Consider any n-bay folded tree. Every tree output Q is on a chain C containing, for each label P_i, one and only one vertex labeled P_i. Hence, there is a tree state S in which all the wires of C (including Q) are in state 1. Also, for any different tree state S', there will be at least one vertex of C in a different setting and so Q will be in state 0 in S'. Therefore, (1) each tree output will be in state 1 in one and only one tree state. Furthermore, (2) any two tree outputs will be in state 1 in different tree states, which can be proved as follows. If Q_1 and Q_2 are different tree outputs on chains C_1 and C_2, respectively, then let i be the number of the latest bay in which C_1 and C_2 have a vertex V in common. Suppose that V is labeled P_k, and suppose (without loss of generality) that C_1 includes the upper vertex-output of V. C_2 must then include the lower vertex-output of V. In order for Q_1 to be in state 1, V will have to be in the upper setting, and, in order for Q_2 to be in state 1, V will have to be in the lower setting. Hence, Q_1 and Q_2 are in state 1 in different tree states.

Since there are as many tree states as tree outputs it follows from (1) and (2) that the tree is complete decoding. This completes our proof of Theorem 1.

An *n-bay standard tree* is an n-bay labeled-tree in which any two vertices have the same label if and only if they are in the same bay. Figure 5 is a 3-bay standard tree. (If we interpret the vertices as in Fig. 2, an n-bay standard tree is essentially the same as the 2-n tree defined in Section 4 of (2).)

Consider now two electrical devices, T_1 realizing a 6-bay standard tree and T_2 realizing a 6-bay folded tree in which the number of vertices labeled $P_1, \ldots, P_6$, respectively, is 1, 12, 12, 12, 13, 13. Both perform the same decoding function but the latter has a more nearly equal distribution of labels. Since all of the vertices with the same label are set in the same way at any one time, this means that the load distribution on the wires controlling the settings of the vertices is

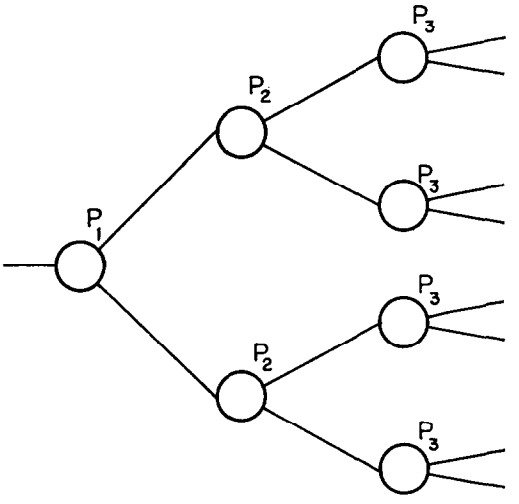

FIG. 5.

more nearly equal in T_2 than in T_1. (The sequence 1, 12, 12, 12, 13, 13 will later be referred to as the "loading sequence" of the tree realized by T_2.) If the devices are constructed of relays (and for reasons indicated earlier in this section the tree is more important for relay contact nets than for electronic digital computing circuits), a further factor needs to be considered. This factor is the number of contacts that each available coil controls. One transfer contact can realize one vertex, but two contacts of the same relay cannot realize vertices with different labels. If one had available relays with eight transfer contacts each, T_2 with load distribution 1, 12, 12, 12, 13, 13 would require eleven relays, while the (standard tree) circuit T_1 would require only ten relays. On the other hand, a (folded tree) circuit with load distribution 1, 8, 8, 15, 15, 16 would require only nine relays. Depending upon what physical equipment is available, different load distributions

for a folded tree offer greater or less advantage. It is therefore of practical interest to know what different load distributions are possible for folded trees.

We are thus led to consider the interesting problem of determining how the P_i's can, in general, be distributed among the $2^n - 1$ vertices of a folded tree. More precisely, given a sequence of n integers $a_1, \ldots, a_n$, is there an n-bay folded tree having, for each i, a_i vertices labeled P_i? A condition for a sequence of positive integers to have a folded tree corresponding to it was stated in (1) and there proved to be sufficient. That condition is the property of "admissibility" as defined in the following section. All the sequences discussed in the preceding examples have this property; 1, 2, 5, 6, 20, 29 is a sequence that does not. In Section 3 of this paper we prove admissibility to be a necessary condition also; and in Sections 4 and 5 we present and justify a method of construction which when applied to any admissible sequence of positive integers results in a folded tree corresponding to that sequence, thus providing a proof of sufficiency different from Shannon's earlier proof which was not constructive.

3. LOADING SEQUENCES; ADMISSIBILITY

In this section we define an arithmetic condition on sequences of non-negative integers, which we shall call the condition of "admissibility," and show it to be a necessary condition for a sequence to represent a distribution of labels among the vertices of a folded tree. To define "admissibility" we must first introduce some additional notions.

Let $V(i,j)$, $1 \leqq j \leqq 2^{i-1}$, be the jth vertex in the ith bay. Inasmuch as the bays are vertical in our figures, $V(i,1)$ is the top vertex of the ith bay, $V(i,2)$ the vertex next to the top, etc. It is clear that the vertex-outputs of $V(i,j)$ are the vertex-inputs of $V(i+1,2j-1)$ and $V(i+1,2j)$. Now let us consider all the chains which include $V(i,j)$. These all have a common vertex in each of the first i bays. That part of the labeled-tree which includes $V(i,j)$ and all vertices from the last $n - i$ bays which are on chains containing $V(i,j)$, together with all vertex-inputs and vertex-outputs of those vertices, is itself a labeled-tree and is called a *minor tree*. We refer to it as $T(i,j)$ since it is determined by $V(i,j)$. $T(1,1)$ is, of course, the original labeled-tree itself. Note Fig. 6 in which $T(2,2)$ is circled.

We assume without any loss of generality that the m labels of $T(1,1)$ are $P_1, P_2, \ldots, P_m$. The *index* of P_k in $T(i,j)$ is the number of vertices labeled P_k in $T(i,j)$ which we denote by $a_k(i,j)$. The *loading sequence* $S(i,j)$ of $T(i,j)$ is the sequence $a_1(i,j), \ldots, a_m(i,j)$. Note that $a_k(i,j)$ is sometimes 0. $S(1,1)$ is then the loading sequence of the original labeled-tree $T(1,1)$. In Fig. 6, $S(2,2)$ is 2, 2, 1, 2, while $S(1,1)$ is 3, 5, 3, 4. Note that $T(i,j)$, $V(i,j)$, $S(i,j)$, and $a_k(i,j)$ are functionally dependent upon the labeled-tree under consideration as well as upon

i and j, even though that functional dependence is not explicit in our notation.

Where S is a finite sequence of non-negative integers $a_1, \ldots, a_m$, we define $M(S)$ to be the sequence rearranged in monotonic non-decreasing order, zeros deleted. Thus, if S is 3, 7, 5, 0, 2, 1, 0, 5, $M(S)$ is 1, 2, 3, 5, 5, 7.

A sequence S of m integers satisfies the *unit condition* if there is a 1 in the sequence. S satisfies the *total sum condition* if the sum of the terms of S is $2^p - 1$, where p is the number of nonzero terms. S

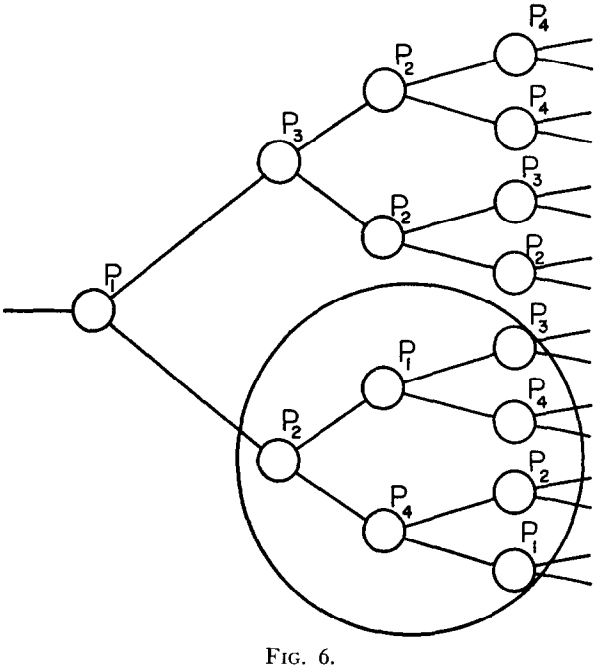

FIG. 6.

satisfies the *partial sum condition* if, for each $k \leqq p$, the sum of the first k terms of $M(S)$ is $\geqq 2^k - 1$. A sequence S is *admissible* if it satisfies all of these three conditions. Thus, 0, 5, 0, 1, 0, 5, 4 is an admissible sequence in which $m = 7$ and $p = 4$; and 1, 9, 5, 10, 6 is an admissible sequence in which $m = p = 5$.

Theorem 2: A minor tree $T(i,j)$ of an n-bay folded tree is an $(n - i + 1)$-bay folded tree.

Proof: Obviously $T(i,j)$ has $n - i + 1$ bays. All the chains of $T(1,1)$ containing $V(i,j)$ have the vertices of the first i bays in common. These are labeled with exactly i of the labels. Now each chain must contain each of the remaining $n - i$ labels in the last $n - i$ bays. But these chains (with the vertices of the first $i - 1$ bays deleted) are the chains of $T(i,j)$. Hence, $T(i,j)$ is a folded tree.

Theorem 3: In an n-bay folded tree $S(i,j)$ has $i - 1$ zeros and $n - i + 1$ non-zero terms.

The proof is immediate from Theorem 2.

Theorem 4: In a folded tree, if $V(i,j)$ is labeled P_k, then $a_k(i,j) = 1$.

Proof: Since $V(i,j)$ is labeled P_k, no other vertex in any chain containing $V(i,j)$ is labeled P_k. For suppose that $V(i',j')$, $i' \neq i$, on a chain C with $V(i,j)$ is also labeled P_k. Then C has $n - 2$ other vertices which must (in order to satisfy the folded tree condition) be labeled with $n - 1$ other labels which is impossible.

Theorem 5: In a folded tree $S(i,j)$ satisfies the unit condition.

Proof: This follows directly from Theorem 4.

Theorem 6: In a folded tree $S(i,j)$ satisfies the total sum condition.

Proof: This follows from Theorem 3 since there are $2^{n-i+1} - 1$ vertices in $T(i,j)$.

Theorem 7: In an n-bay folded tree $a_k(i + 1,2j - 1) = 0$ if and only if $a_k(i + 1,2j) = 0$.

Proof: $T(i,j)$ by Theorem 2 is a folded tree and, therefore, each P_k which labels any vertex of $T(i,j)$ has to label at least one vertex in each chain of $T(i,j)$. Every such chain contains $V(i,j)$. Beyond $V(i,j)$ each chain of $T(i,j)$ lies wholly within $T(i + 1,2j - 1)$ or $T(i + 1,2j)$. Hence any label occurs in $T(i + 1,2j - 1)$ if and only if it occurs in $T(i + 1,2j)$, from which Theorem 7 follows directly.

Theorem 8: In an n-bay folded tree $S(i,j)$ satisfies the partial sum condition.

Proof: The proof is by induction, beginning at the nth bay of the tree and going to the left. We first show (1) that, for every $j \leqq 2^{n-1}$, the theorem holds for $S(n,j)$. We then show (2) that, if it holds for $S(i + 1,j)$ for every $j \leqq 2^i$, it holds for $S(i,j)$ for every $j \leqq 2^{i-1}$. (1) is true by Theorem 3 for $S(n,j)$ has only one non-zero term. We prove (2) by showing that, if it holds for $S(i + 1,2j - 1)$ and $S(i + 1,2j)$, then it holds for $S(i,j)$. Suppose $V(i,j)$ is labeled P_q. Let $M(S(i,j))$ be $a_q(i,j)$, $a_{g_1}(i,j)$. $a_{g_2}(i,j)$, . . . , $a_{g_{n-i}}(i,j)$, where, of course, $a_q(i,j) = 1$. By hypothesis $S(i + 1,2j - 1)$ and $S(i + 1,2j)$ satisfy the partial sum condition; in other words, for each $k \leqq n - i$, the sum of the first k terms of $M(S(i + 1,2j - 1))$ or $M(S(i + 1,2j)) \geqq 2^k - 1$. Now, if such a condition holds for a monotonic non-decreasing sequence, it holds for the sequence in any order. Furthermore, since, for $k \neq q$, $a_k(i,j) = a_k(i + 1,2j - 1) + a_k(i + 1,2j)$, with the aid of Theorem 7 it is

easy to see that the non-zero terms of $S(i+1,2j-1)$ are $a_{g_1}(i+1, 2j-1), \ldots, a_{g_{n-i}}(i+1,2j-1)$ and the non-zero terms of $S(i+1, 2j)$ are $a_{g_1}(i+1,2j), \ldots, a_{g_{n-i}}(i+1,2j)$. Hence for each $k \leqq n-i$, $\sum_{x=1}^{k} a_{g_x}(i+1,2j-1) \geqq 2^k - 1$ and $\sum_{x=1}^{k} a_{g_x}(i+1,2j) \geqq 2^k - 1$. Hence,

$$\sum_{x=1}^{k} (a_{g_x}(i+1,2j-1) + a_{g_x}(i+1,2j)) \geqq 2^{k+1} - 2.$$ Clearly,

$$a_q(i,j) + \sum_{x=1}^{k} a_{g_x}(i,j) = 1 + \sum_{x=1}^{k} (a_{g_x}(i+1,2j-1) + a_{g_x}(i+1,2j));$$

hence for any $k \leqq n-i$, $a_q(i,j) + \sum_{x=1}^{k} a_{g_x}(i,j) \geqq 2^{k+1} - 1$, which means that the sum of the first $k+1$ terms of $M(S(i,j))$ is $\geqq 2^{k+1} - 1$, proving that $S(i,j)$ satisfies the partial sum condition.

Theorem 9: The loading sequence $S(1,1)$ of an n-bay folded tree is an admissible sequence of n non-zero terms.

This result follows immediately from Theorems 3, 5, 6, and 8, putting $i = j = 1$.

4. THE SPLITTING TECHNIQUE

In this section we provide an effective method of constructing a folded tree with a given admissible loading sequence of positive integers. The proof that this method will produce a folded tree is given in the next section. The procedure consists in taking the loading sequence proposed for the folded tree $T(1,1)$, selecting the unit term for $V(1,1)$, and dividing the remaining terms so as to yield two admissible sequences, one for $T(2,1)$ and the other for $T(2,2)$. This procedure, called the "splitting technique," is then repeated for each of those two sequences to provide admissible sequences for $T(3,1)$, $T(3,2)$, $T(3,3)$, and $T(3,4)$. The process is iterated until $T(n,2^{n-1})$ is reached. It will be proved in the next section that each vertex is thus provided with a label and that, for each k, the proper number of vertices will be labeled P_k.

It must be admitted that there are alternative splitting techniques which applied to the same admissible sequence give different folded trees. Our splitting technique, for example, applied to the admissible sequence 1, 4, 5, 5 gives a folded tree having three different labels in its last bay, but there is also a folded tree with the same loading sequence which has only two different labels in its last bay (see Fig. 7).

In this section and in Section 5 we let $S'(i,j)$, which is $a_1'(i,j)$, $a_2'(i,j), \cdots, a_n'(i,j)$, be the sequence obtained by repeated applications of the splitting technique from $S'(1,1)$, which is an arbitrary admissible

sequence without zero terms. The symbols $S'(i,j)$ and $S(i,j)$ are to be clearly distinguished even though in some cases they refer to the same sequence of numbers. $S'(i,j)$ is the sequence derived by our splitting technique from a given admissible sequence $S'(1,1)$, whereas $S(i,j)$ is the loading sequence of $T(i,j)$, a minor tree of a given labeled-tree.

Let us suppose that $S'(i,j)$ is a given admissible sequence. We describe now the *splitting technique* which determines $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$. For every k, if $a_k'(i,j) = 1$ or 0, then let $a_k'(i + 1,2j - 1) = a_k'(i + 1,2j) = 0$. Let $M(S'(i,j))$ be $1,d_1, \cdots, d_p$. (It will turn out that $p = n - i$.) We then determine sequences $b_1, \cdots, b_p$ and $c_1, \cdots, c_p$, such that, if d_x is $a_k'(i,j)$, b_x and c_x are to be $a_k'(i + 1,2j - 1)$ and $a_k'(i + 1,2j)$, respectively. The recursive procedure for determining b_x and c_x is as follows: (1) put $c_1 = 1$ and

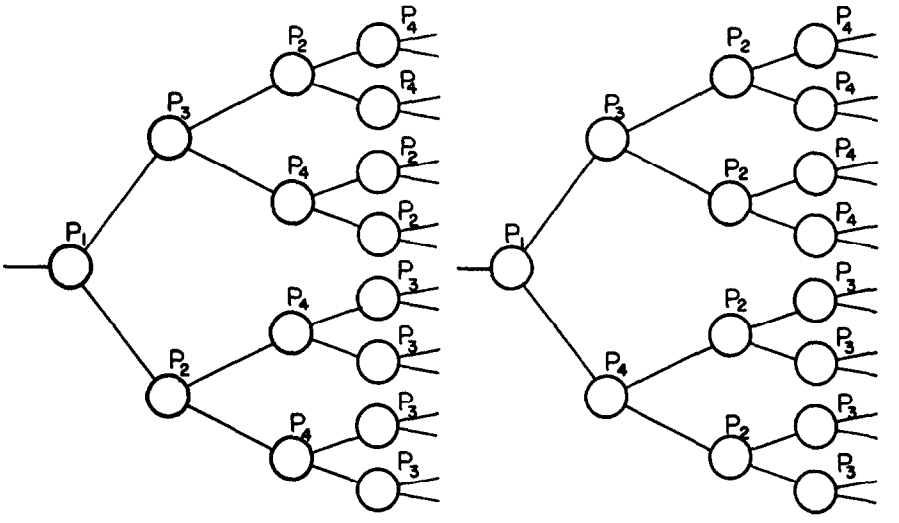

FIG. 7.

$b_1 = d_1 - 1$; (2) if $b_1 = 1$ (that is, if $d_1 = 2$), proceed to (3), otherwise first put $b_2 = 1$ and $c_2 = d_2 - 1$; (3) (for each x, $x \geqq 3$, and, for $x = 2$ if $d_1 = 2$) assuming that $b_1, \cdots, b_{x-1}$ and $c_1, \cdots, c_{x-1}$ have been determined, put

$$b_x = \left[\frac{d_x + \Delta_x}{2} \right]$$

and

$$c_x = \left[\frac{d_x + 1 - \Delta_x}{2} \right],$$

where

$$\Delta_x = \sum_{y=1}^{x-1} (c_y - b_y).$$

Here $[\phi]$, for any real number ϕ, is the integral part of ϕ, that is, the greatest integer not greater than ϕ. The reader can readily verify that for each x, $b_x + c_x = d_x$.

In actual calculation the work is even simpler than it appears from the formal statement of the procedure. The idea behind the splitting technique is simply to provide for the 1 in each sequence, and, for each d_x, to find a b_x and c_x such that $b_x + c_x = d_x$, in such a way that

$$\sum_{y=1}^{x} b_y$$

is as nearly equal as possible to

$$\sum_{y=1}^{x} c_y.$$

We also require that

$$\sum_{y=1}^{x} c_y$$

be never less than

$$\sum_{y=1}^{x} b_y,$$

except where $d_1 > 2$ and $x = 2$. Thus, all terms except possibly the first three will be split as evenly as possible so that Δ_x for $x > 3$ is either 0 or 1. For $x > 3$, then, if d_x is even, $b_x = c_x = d_x/2$; if d_x is odd and Δ_x is 1, then $b_x = (d_x + 1)/2$ and $c_x = (d_x - 1)/2$; if d_x is odd and Δ_x is 0, then $b_x = (d_x - 1)/2$ and $c_x = (d_x + 1)/2$. Furthermore, Δ_x does not have to be recalculated each time for $x > 3$; for if d_x is even, then $\Delta_{x+1} = \Delta_x$, and if d_x is odd, then Δ_{x+1} is 1 if Δ_x is 0 and 0 if Δ_x is 1.

The following table carries through a calculation from $S'(i,j)$ to $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$.

$S'(i,j)$	0	3	0	1	8	14	5
d-sequence	3	5	8	14			
b-sequence	2	1	5	7			
c-sequence	1	4	3	7			
$S'(i + 1,2j - 1)$	0	2	0	0	5	7	1
$S'(i + 1,2j)$	0	1	0	0	3	7	4

It is simple to rearrange the terms of the b and c sequences in forming the sequences $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$ in accordance with the rule that where d_x is $a_k'(i,j)$, b_x and c_x are $a_k'(i + 1,2j - 1)$ and $a_k'(i + 1,2j)$, respectively.

To illustrate the case where $d_1 = 2$, consider the $M(S'(i,j))$ sequence 1, 2, 5, 9, 14. Here the b-sequence is 1, 2, 5, 7 and the c-sequence is 1, 3, 4, 7.

Since we have shown how an admissible $S'(i,j)$ is split into $S'(i + 1,2j - 1)$ and $S'(i + 1,2j)$, it is easy to see that given $S'(1,1)$ all the $S'(i,j)$ can be calculated, provided all the applications of the

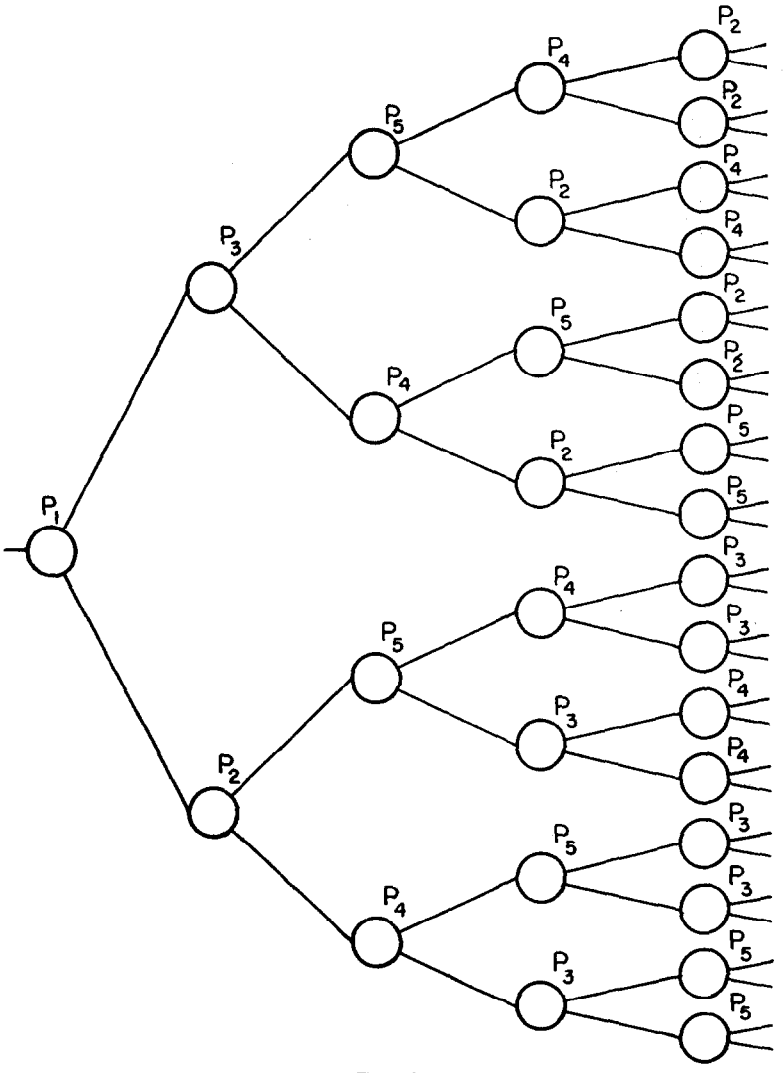

FIG. 8.

splitting technique yield admissible sequences. Once all these have been calculated, a folded tree whose $S(i,j)$ is $S'(i,j)$ can easily be constructed. If there are n terms in $S'(1,1)$, then, of course, we want an n-bay folded tree. An n-bay tree is constructed and each vertex

$V(i,j)$ is labeled P_k where $a_k'(i,j)$ is unity. (In the next section we prove that a vertex will be assigned one and only one label by this procedure.)

We will now provide an example showing the use of this procedure in constructing a 5-bay folded tree with the given admissible loading sequence, 1, 7, 7, 8, 8, which is one of the most evenly balanced loading sequences possible for a 5-bay folded tree. We omit the d-sequence, as well as the b-sequence and c-sequence, in each step.

Since $S'(1,1)$ is 1, 7, 7, 8, 8,

$S'(2,1)$ is 0, 6, 1, 4, 4, and $S'(2,2)$ is 0, 1, 6, 4, 4;
$S'(3,1)$ is 0, 3, 0, 3, 1, and $S'(3,2)$ is 0, 3, 0, 1, 3;
$S'(3,3)$ is 0, 0, 3, 3, 1, and $S'(3,4)$ is 0, 0, 3, 1, 3;
$S'(4,1)$ is 0, 2, 0, 1, 0, and $S'(4,2)$ is 0, 1, 0, 2, 0;
$S'(4,3)$ is 0, 2, 0, 0, 1, and $S'(4,4)$ is 0, 1, 0, 0, 2;
$S'(4,5)$ is 0, 0, 2, 1, 0, and $S'(4,6)$ is 0, 0, 1, 2, 0;
$S'(4,7)$ is 0, 0, 2, 0, 1, and $S'(4,8)$ is 0, 0, 1, 0, 2.

For each i,j we label $V(i,j)$ P_k where the kth term of $S'(i,j)$ is unity. It is not necessary to compute $S'(5,j)$ for any j, since every $V(5,j)$ can be labeled with that label which has not already appeared in the chain containing that $V(5,j)$. The result is Fig. 8.

(To be concluded.)

REFERENCES

(1) C. E. SHANNON, "The Synthesis of Two-Terminal Switching Circuits," *Bell System Technical Journal*, Vol. 28, pp. 59–98, January, 1949.

(2) A. W. BURKS, R. MCNAUGHTON, C. H. POLLMAR, D. W. WARREN AND J. B. WRIGHT, "Complete Decoding Nets: General Theory and Minimality," *J. Soc. Industrial and Applied Math.*, Vol. 2, No. 4, pp. 201–243 (1954).

(3) W. KEISTER, A. E. RITCHIE AND S. H. WASHBURN, "The Design of Switching Circuits," New York, 1951.

(4) A. W. BURKS, C. H. POLLMAR, D. W. WARREN AND J. B. WRIGHT, "Minimal Switch Theory and the Folded Tree," Vol. III, *Language Conversion For Digital Computers*, ERI Project M828, Ann Arbor, 1 March 1953. (Copies may be obtained by writing Mr. George W. Patterson, Burroughs Corporation, Research Center, Paoli, Pennsylvania.)