Journal
of
The Franklin Institute
Devoted to Science and the Mechanic Arts

Vol. 257 MARCH, 1954 No. 3

THE SYNTHESIS OF SEQUENTIAL SWITCHING CIRCUITS *

BY

D. A. HUFFMAN[1]

ABSTRACT

An orderly procedure is developed by which the requirements of a sequential switching circuit (one with memory) can be reduced to the requirements of several combinational switching circuits (those without memory). Important in this procedure are:

1. the flow table: a tabular means by which the requirements of a sequential switching circuit may be stated precisely and by which redundancy in these requirements may be recognized and eliminated, and
2. the transition index: a new variable which indicates the stability (or lack of stability) of a switching device.

The role of those switching devices which are not directly controlled by the input of a sequential switching circuit is investigated thoroughly. The resulting philosophy, which is exploited in synthesis procedures for circuits using either relay or vacuum-tube switching devices, is valid for circuits using other devices as well.

PART I †

I. INTRODUCTION TO RELAY CIRCUIT THEORY

Historical Background

In 1938, C. E. Shannon established an orderly algebraic procedure for the treatment of relay contact networks.[2] This major theoretical advance was based on an analogy with the calculus of propositions used in symbolic logic. In 1948, G. A. Montgomerie[3] described a concise

* This paper is derived from a dissertation submitted in partial fulfillment of the requirements for the degree Doctor of Science at the Massachusetts Institute of Technology.

[1] Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass.

† Part II will be published in this JOURNAL for April, 1954.

[2] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Trans. AIEE*, Vol. 57, pp. 713–723 (1938).

[3] G. A. Montgomerie, "Sketch for an Algebra of Relay and Contactor Circuits," *Jour. IEE*, Vol. 95, Part III, pp. 303–312 (1948).

method of specifying static (or "combinational") relay circuits which has since been used extensively by Shannon [4] and others, and is now called a "table of combinations." The need for similar orderly analysis methods for circuits with dynamic relay action was realized by Montgomerie,[3] and some of the matrices described here bear superficial resemblance to his, even though they were developed independently.

Assumed Form of the Switching Circuit

Our generalized relay switching circuit will have p input terminals, and q output terminals. Each of the input terminals will be connected to a terminal (called the *control terminal*) of the coil winding of a single-winding relay; the other end of the winding will be connected to the proper supply voltage. (See Fig. 1, for example.) These relays, which are under the direct control of the circuit inputs, are designated as X or *primary* relays. All other relays (if any) in the circuit are to be called Y or *secondary* relays. Their number will be designated s.

Each of the s secondary relays will be controlled by a network of contacts from (in general) all of the relays—both primary and secondary—in the circuit. Similar networks of contacts (the Z networks) will lead to each of the q output terminals from ground.

Contacts associated with a given relay (R) will be given the lowercase designations. *Normally open* contacts are labeled with the *unprimed* letter (r), and the *normally closed* contacts with the *primed* letter (r'). We shall assume that, for a given relay:

1. All normally open contacts are open (or closed) at the same time. Likewise, all normally closed contacts are open (or closed) simultaneously.

2. When the normally open contacts for a given relay are open (or closed), the normally closed contacts are closed (or open), and vice versa.

Algebraic Description of the Relays

A *binary* (two-valued) variable is one which may, at any given time, assume just one of two possible *complementary* values. The binary variables used in describing a relay circuit represent the absence or presence of a metallic path. These variables are assigned the values zero and one. A contact, or a network of contacts, has a *transmission of unity* when it is *closed*, and a *zero transmission* when it is open.[5]

In terms of the transmission concept, $r = 0$ will be interpreted to mean: The normally open contacts on the relay R are open; that is, the

[4] C. E. Shannon, "The Synthesis of Two-Terminal Switching Circuits," *Bell System Tech. Jour.*, Vol. 28, pp. 59–98 (1949).

[5] Transmission is thus somewhat analogous to admittance. The dual concept, that of *hindrance*, is somewhat analogous to impedance; a closed circuit has a zero hindrance and an open circuit has unity hindrance. We shall use the idea of transmission exclusively.

relay is *unoperated.* Similarly $r = 1$ will mean: The normally open contacts on R are closed; that is, the relay is *operated.* Of course—with the assumptions made previously—whenever $r = 0$, then $r' = 1$ and whenever $r = 1$, then $r' = 0$. Thus r and r', when considered as contact variables, always have complementary values.[6]

A relay is *energized* or *de-energized* according as the transmission of the contact network in series with its winding has a transmission of unity or zero, respectively. It will be convenient to use the upper-case letter R as the notation for the transmission of this network. Thus, when $R = 1$, the relay is energized; and when $R = 0$, the relay is de-energized.

It is necessary to make a fine distinction between the state of operation of a relay and its state of energization, as reflected by the variables r and R, respectively. If r and R have the same value the relay is in a *stable* state. If instead, r and R have complementary values the relay is in an unstable state. Since both r and R may each have only the values zero and one, there are just four mutually exclusive situations which may arise:

1. $r = 0$ and $R = 0$; a stable state.
2. $r = 0$ and $R = 1$; an unstable state; the relay is unoperated and is energized. If R continues to have the value one, then eventually the relay will become operated and the value of r will change from zero to one. The time required is called the relay *operate* time.
3. $r = 1$ and $R = 0$; an unstable state; the relay is operated and is de-energized. If R continues to have the value zero, then eventually the relay will become unoperated and the value of r will change from one to zero. The time required is called the relay *release* time.
4. $r = 1$ and $R = 1$; a stable state.

We now define a new variable, τ_R, called the *transition index* which reflects the stability or instability of a relay R. When $\tau_R = 0$ the relay is stable; when $\tau_R = 1$ the relay is unstable.

For convenience we will make use of an algebraic operation which we shall call cyclic addition,[7] and for which the notation is $\oplus$. It will have the following properties:

$$0 \oplus 0 = 1 \oplus 1 = 0; \qquad 0 \oplus 1 = 1 \oplus 0 = 1. \qquad (1)$$

In terms of this notation

$$\tau_R = r \oplus R \qquad \text{and} \qquad R = r \oplus \tau_R. \qquad (2\text{-}a, b)$$

[6] The priming notation may be used whenever the idea of complementation exists. Thus, $0' = 1$ and $1' = 0$.

[7] In the language of congruences, τ_R is congruent, modulo two, to the sum of r and R. See, for example, Birkoff and MacLane, "A Survey of Modern Algebra," New York, The Macmillan Company, 1941, pp. 23–29.

Combinational and Sequential Circuits

Consider for the time being a switching circuit which has no secondary relays; for example, that of Fig. 1(*a*). For this kind of circuit the contact networks which lead to the output terminals must be composed of contacts from relays which are under the direct influence of the input (the primary relays). Since each primary relay may be either operated or unoperated, there are 2^p possible states of operation for the p primary relays, taken collectively. Each of these *primary states* determines uniquely a transmission of either zero or unity at each output terminal. In Fig. 1(*a*), for instance, there is an output ground if X_1 is operated or if X_2 is unoperated, or both. If X_1 is unoperated and if X_2 is operated, however, the output will be ungrounded.

Consider now a circuit having secondary relays as well as primary relays—for example that of Fig. 1(*b*). Assume that all three of the relays are unoperated. Operation of the X_2 relay has no effect on the Y relay but does result in a ground at the output terminal (since the y' contact is closed). As long as Y remains unoperated, the output is grounded or not grounded according to whether X_2 is or is not operated.

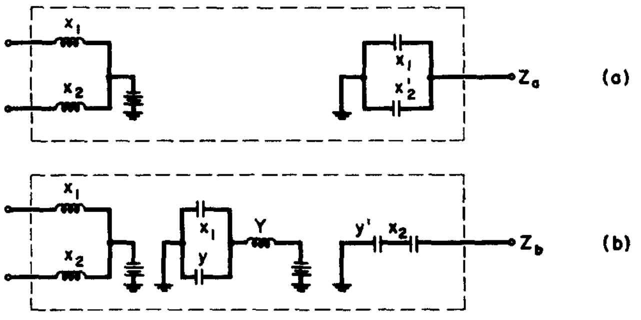


FIG. 1. Illustrating combinational and sequential switching circuits.

Let us now assume that the X_1 relay is operated. The closing of its normally open contact, x_1, energizes and subsequently operates the Y relay, and closes the y contact. Now the state of operation of X_1 cannot affect Y since the latter is permanently energized through its own normally open contact. (Permanently, that is, until the supply voltages are removed.) Now the y' contact in the output network is open and operation of X_2 cannot ground the output terminal.

Notice that the output transmission is not a unique function of the primary relay states of operation but depends as well on the past history of the circuit. The circuit may be said to have a *memory;* it "remembers" whether or not X_1 has been operated.

We may generalize from these two simple examples: In a circuit having no secondary relays there can be no "memory"; the states of operation of the primary relays uniquely determine the output transmissions. Such a circuit is called a *combinational* circuit. In a circuit having secondary relays, the possibility of a "memory" exists since the states of operation may not uniquely determine the output transmissions. A circuit having secondary relays will be called a *sequential* circuit.

Specification of the Terminal Characteristics of Switching Circuits

The terminal characteristics of a combinational circuit may be described equally well by specifying the nature of the contact networks at the outputs. A useful means of specification of these networks (or of *any* two-terminal contact network) is the *table of combinations.* In it are listed the 2^p possible (collective) states of operation of the p primary relays which may contribute contacts to such an output network; beside each of these 2^p states is listed the transmission of the network. In Table I, the table of combinations tells us that the Z_a network (Fig. 1(*a*)) has a transmission of zero if and only if X_1 is unoperated and X_2 is operated.

TABLE I.—*Tables of Combinations for the Networks in Fig. 1.*

(*a*)			(*b*)			(*c*)		
x_1	x_2	Z_a	x_1	y	Y	x_2	y	Z_b
0	0	1	0	0	0	0	0	0
0	1	0	0	1	1	0	1	0
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

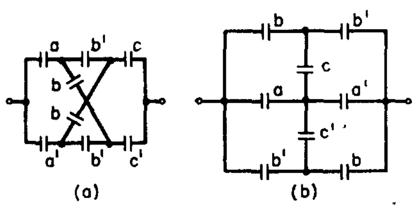
The precise specification of a sequential circuit is more difficult. We can list the transmissions of the Y and Z networks in tables of combinations (see Table I(*b*) and (*c*)) but this listing tells us little about the terminal characteristics of the circuit or what function "memory" plays in its operation.

Up to the present time no precise way of tabulating the terminal characteristics of the general sequential switching circuit has been published. The *flow table* developed in Section II is the author's answer to this problem. Without an exact specification, such as the one the flow table gives, we cannot hope to be able to lay down rules for the synthesis of the general sequential circuit.

Comments on the Synthesis of Contact Networks

Our synthesis method for sequential circuits will lead to the listing of the transmission requirements for several Y and Z networks in tables of combinations. The physical realization of contact networks which meet the requirements in a table of combinations has received much study and there is much research yet to be done. We will not attempt to show how to design such networks but will usually diagram one possible physical realization and the reader may verify that it corresponds to its table of combinations.[8]

Part of the problem of synthesizing contact networks exists because several quite different-appearing networks may be terminally equivalent. It is not always easy to decide on a criterion of merit for these different networks or to be sure that once a "good" network is found a "better" one does not also exist. However, a common denominator for all equivalent contact networks is the table of combinations. For example, Fig. 2 gives two equivalent networks along with the table of combinations which they have in common.



a	b	c	Network Transmission
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(c)

Fig. 2. Two equivalent networks and their table of combinations.

II. THE ANALYSIS OF SEQUENTIAL RELAY CIRCUITS

Matrix Representation for Two-Terminal Networks

In each of the relay switching circuits discussed in this paper, there is a two-terminal contact network connected between ground and the control terminal of each secondary relay. These networks have been given the notations $Y_1, Y_2, \cdots, Y_s$. In addition, there are two-terminal networks between ground and each of the output terminals. These networks are to be given the notations $Z_1, Z_2, \cdots, Z_q$. Each of the Y and Z networks may be a function of all the contact variables for the circuit—that is, of the $(p + s)$ variables $x_1, x_2, \cdots, x_p, y_1, y_2, \cdots, y_s$. Therefore, each of these transmission functions may be represented in a table of combinations with rows corresponding to all possible combinations of the $(p + s)$ contact variables. It is more convenient, however, to use modified tables of combinations which, for the sake of compactness, are put in the form of rectangular matrices.

To illustrate the construction of such matrices, Table II shows the tables of combinations for the three secondary relay control networks

[8] For further study on contact networks the reader is referred to W. Keister, A. E. Ritchie and S. Washburn, "The Design of Switching Circuits," New York, D. Van Nostrand Company, Inc., 1951.

For recent papers on the reduction of combinational circuit requirements to simple forms, refer to M. Karnaugh, "The Map Method for Synthesis of Combinational Logic Circuits," AIEE Technical Paper No. 53-217, April, 1953; or to E. E. Veitch, "A Chart Method for Simplifying Truth Functions," *Proc. Assn. for Computing Machinery*, May, 1952; or to W. H. Burkhart, "Theorem Minimization," *ibid.*

TABLE II.—*Tables of Combinations for the Y and Z Networks of Fig. 3.*

x_1	x_2	y_1	y_2	y_3	Y_1	Y_2	Y_3	Z	
0	0	0	0	0	0	0	0	0	
0	0	0	0	1	1	0	1	0	
0	0	0	1	0	1	1	0	0	
0	0	0	1	1	0	1	1	0	
0	0	1	0	0	0	0	0	0	
0	0	1	0	1	1	0	1	1	*
0	0	1	1	0	1	1	0	0	
0	0	1	1	1	0	1	1	0	
0	1	0	0	0	1	0	0	0	
0	1	0	0	1	1	0	0	0	
0	1	0	1	0	0	0	0	0	
0	1	0	1	1	0	0	0	0	
0	1	1	0	0	1	0	0	0	
0	1	1	0	1	1	0	0	0	
0	1	1	1	0	1	1	1	0	
0	1	1	1	1	1	1	1	0	
1	0	0	0	0	0	1	0	0	
1	0	0	0	1	0	0	1	0	
1	0	0	1	0	0	1	0	0	
1	0	0	1	1	0	0	1	0	
1	0	1	0	0	1	0	0	0	
1	0	1	0	1	1	0	0	0	
1	0	1	1	0	1	0	0	0	
1	0	1	1	1	1	0	0	0	
1	1	0	0	0	1	0	0	0	
1	1	0	0	1	1	0	0	0	
1	1	0	1	0	0	0	0	0	
1	1	0	1	1	0	0	0	0	
1	1	1	0	0	1	0	0	0	
1	1	1	0	1	1	0	0	0	
1	1	1	1	0	1	0	0	0	
1	1	1	1	1	1	0	0	0	

and the single output network of the circuit of Fig. 3. Table III gives the corresponding matrix forms. In general, the Y matrix will have as entries ordered s-tuples of the proper values of $Y_1, Y_2, \cdots, Y_s$; and similarly, the Z matrix will have as its entries ordered q-tuples of the proper values of $Z_1, Z_2, \cdots, Z_q$. Positions in the matrix will be given in the form $(x;y)$, where x is the ordered p-tuple of the values of the primary contact variables $x_1, x_2, \cdots, x_p$, and where y is the ordered s-tuple of the values of the secondary contact variables $y_1, y_2, \cdots, y_s$. For example, the entry "1" in the (00;101) position of the Z matrix of

Table III(b) corresponds to the starred row of Table II. In this same starred row the values for Y_1, Y_2, and Y_3 are one, zero, and one, respectively. And so the entry in the (00;101) position of the Y matrix of Table III(a) is "101."

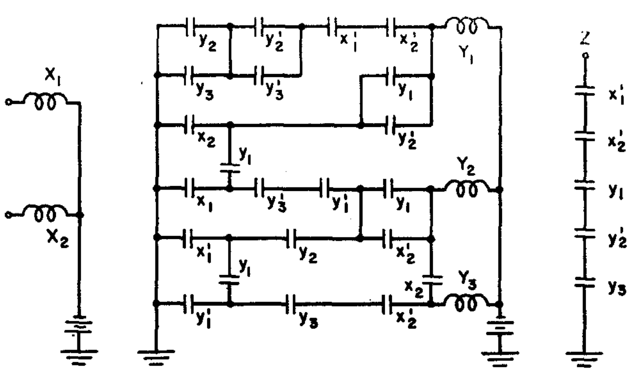


FIG. 3. A sequential switching circuit.

TABLE III.—*Y and Z Matrices Formed from the Data of Table II.*

(a)
The Y Matrix

y: \ x:	00	01	10	11
000	000	100	010	100
001	101	100	001	100
010	110	000	010	000
011	011	000	001	000
100	000	100	100	100
101	101	100	100	100
110	110	111	100	100
111	011	111	100	100

(b)
The Z Matrix

y: \ x:	00	01	10	11
000	0	0	0	0
001	0	0	0	0
010	0	0	0	0
011	0	0	0	0
100	0	0	0	0
101	1	0	0	0
110	0	0	0	0
111	0	0	0	0

The Composite Transition Matrix

Next we shall form the *composite transition matrix*, τ, for our present example. We shall make use of the equations

$$\tau_{Y1} = Y_1 \oplus y_1, \quad \tau_{Y2} = Y_2 \oplus y_2, \quad \text{and} \quad \tau_{Y3} = Y_3 \oplus y_3, \qquad (3\text{-}a, b, c)$$

which correspond to Eq. 2(a) applied to the three secondary relays. The mechanics of construction of τ are as follows:

For a given primary relay state, $x = (x_1, x_2, \cdots, x_p)$, and for a given secondary relay state $y = (y_1, y_2, \cdots, y_s)$—that is, for a given *total* relay state $(x;y)$,—there is an entry $(Y_1, Y_2, \cdots, Y_s)$ in the Y matrix. In τ, the proper entry at the $(x;y)$ position is $(\tau_{Y1}, \tau_{Y2}, \cdots, \tau_{Ys})$

$= (Y_1 \oplus y_1, Y_2 \oplus y_2, \cdots, Y_s \oplus y_s)$. For instance, the entry in the (00;010) position of the Y matrix of Table III(a) is "110." Since this entry is in the row corresponding to $y = 010$, then we find the proper entry in the τ matrix of Table IV(a) by adding cyclicly the corresponding components of "110" and "010." The result is "100," and this is inserted in the (00;010) position of the τ matrix. This derived entry is to be interpreted: "For the total relay state (00;010), the relay Y_1 is in an unstable state, but the relays Y_2 and Y_3 are both in stable states."

TABLE IV.—*The τ Matrix Derived from Table III.*

y : \ x :	00	01	10	11
000	000	100	010	100
001	100	101	000	101
010	100	010	000	010
011	000	011	010	011
100	100	000	000	000
101	000	001	001	001
110	000	001	010	010
111	100	000	011	011

Later, in the synthesis process, we shall need to reverse the procedure above and derive the Y matrix from the τ matrix. Then we shall need to make use of the equations

$$Y_1 = \tau_{Y1} \oplus y_1, \qquad Y_2 = \tau_{Y2} \oplus y_2, \cdots, Y_s = \tau_{Ys} \oplus y_s, \tag{4}$$

which correspond to Eq. 2(b) applied to the s secondary relays. In order to derive each entry in the Y matrix, we shall have to add cyclicly the components of the appropriate value of $y = (y_1, y_2, \cdots, y_s)$ to each component of the corresponding entry in τ.

In a τ matrix, each entry consisting entirely of zeros indicates stability for all the secondary relays. If an entry contains a single digit one as a component, then just one secondary relay is in an unstable state, and the entry tells us what the secondary relay state will next be if the input state remains unchanged. For instance, in the τ matrix of our present example, the "100" entry at the (00;010) position indicates that the Y_1 relay is unstable; and thus, if the input state remains "00," the resulting total relay state will be (00;110). A glance at τ reveals that all the secondary relays are stable for this new total relay state, since the entry at this position consists entirely of zeros. The circuit action described above is diagrammed by the heavy arrow in Fig. 4.

An important fact to observe in the use of a composite transition matrix is that a change of state in the secondary relays is indicated by vertical movement in the matrix and in the associated diagram, while a change of input state is accompanied by motion in a horizontal direction. Moreover, once a stable state has been reached for all secondary relays, then further circuit changes can occur only if modification is made in the input state. In other words, if the circuit has a stable secondary relay state, and if the input state is changed from the existing value to another one, then the focal point of our attention must first be moved horizontally into the column corresponding to the new input state; and if further changes are to occur, these must be in the vertical direction.

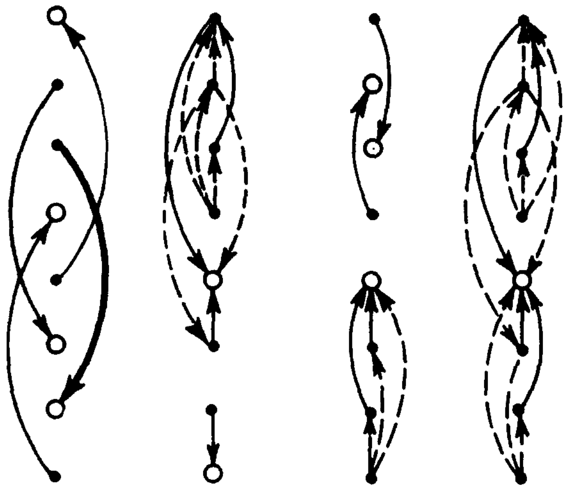


FIG. 4. Transition diagram for the matrix given in Table IV.

Race Conditions

In a composite transition matrix the presence of two or more of the digits "1" indicates that at least two secondary relays are simultaneously unstable, and that a *race condition* exists. In such a case, several secondary relay actions are possible, depending upon the magnitudes of the operating and release times for the relays involved and upon the past history of their excitations. For example, in Table IV, if the total relay state is (10;111), the corresponding entry is "011." This entry indicates that a "race" develops between the relays Y_2 and Y_3 to see which relay will become released first. If Y_2 wins the "race" and is released first, the resulting secondary relay state is "101." If, however, Y_3 is the first relay to release, then the secondary relay state becomes "110." But if a "tie" develops and both relays release at the same time, the secondary relay state will be "100." These three possibilities are indicated by dotted lines rising from the bottom node in the third column of Fig. 4.

All race conditions in Fig. 4 are indicated by dotted lines. In this figure each race condition is *non-critical,* since each of the alternate possibilities leads eventually to the same *ultimate* secondary relay state. For example (see the right-hand column of Fig. 4): If the input state is "11," the ultimate secondary relay state will be "100," no matter what the initial secondary relay state was.

Not all race conditions are of the noncritical variety. It may be that the alternate possibilities present in a race condition lead to different ultimate circuit conditions. Consider, for instance, Table V (and Fig. 5). Here, several ultimate circuit conditions are possible if the total relay state is initially (00;010).

TABLE V.—*A τ Matrix Illustrating Race Conditions and Cycles.*

y: \ x:	00	01	10	11
000	010	000	000	001
001	000	100	011	010
010	101	001	100	110
011	010	010	100	001
100	000	010	000	001
101	000	001	000	010
110	000	100	100	001
111	010	000	100	100

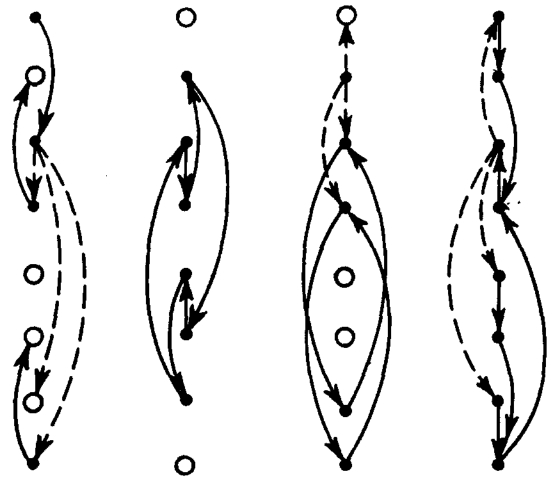


FIG. 5. Transition diagram for τ matrix of Table V.

Cycles and Ultimately Stable Terminal Action

It may be that, for a given input state, internal circuit action is continuously sustained because at least one secondary relay is always maintained in an unstable state. For this situation there will be no single ultimate secondary relay state, but instead there will be a *cycle* of states occurring. The column of Table V (and of Fig. 5) for which the input state is "01" illustrates such a cycle. Here the secondary relay states · · ·, 001, 101, 100, 110, 010, 011, · · · occur in cyclic fashion.

If the output state of a relay switching circuit remains constant for all the secondary relay states of a cycle, we shall say that the circuit has *ultimately stable terminal action.*

It is also possible that race conditions and cycles may coexist in a switching circuit. Two examples of this situation are given in the columns of Table V which correspond to the input states "10" and "11."

The Flow Table

Our attention will now be limited to the analysis of those relay switching circuits in which there are no cycles, and where all race conditions (if any exist) are of the noncritical type. For this kind of circuit we can construct a *flow table, F.* Our running example (Fig. 3, Tables II, III, and IV, and Fig. 4) is of the proper classification, and the associated flow table is given in Table VI.

Table VI.—*The Flow Table Derived from Table IV and Fig. 4.*

y: \ x:	00	01	10	11
000	①	5	3	7
001	8	5	②	7
010	9	5	③	7
011	④	5	2	7
100	1	⑤	⑥	⑦
101	⑧	5	6	7
110	⑨	10	6	7
111	4	⑩	6	7

The circles in F correspond in position to those entries in τ which are made up of zeros only, and which therefore represent those circuit conditions for which all secondary relays are stable. The circles are numbered serially; the order of assignment of the numbers is unimportant. The remaining entries in F are uncircled, and each such entry tells what stable circuit condition will ultimately result if the circuit is put in a total relay state corresponding to the entry in question and if the input state remains unchanged. This stable circuit condition can be derived by analyzing the composite transition matrix, τ. With this rule, and aided by the diagram of Fig. 4, if necessary, we obtain Table VI.

Since we are going to state our synthesis problems in terms of flow tables, it will be worth while to take time to understand thoroughly how a flow table is used and what its entries mean. The flow table can give us a graphic idea of what circuit actions can occur in a relay circuit. Just as in the composite transition matrix, horizontal motion in the flow table corresponds to modification of the input state. Within each row of the table the following rules hold:

1. Each circled entry in a row of a flow table indicates a stable circuit condition, and no further changes will occur unless the input state is modified.
2. Each circled circuit condition within a row of a flow table can lead to any other circuit condition (circled or uncircled) which is listed in the same row of the table.
3. Each uncircled entry in a row of a flow table indicates the stable circuit condition which will ultimately follow if the input state is left constant. This stable circuit condition, circled, will be listed in the same column of the table, but in another row.

For example (see the fifth row of Table VI): If the existing circuit condition is denoted by "6," then circuit conditions "1," "5," or "7" can follow, depending upon whether the input state is changed from "10" to "00," "01," or "11," respectively. If, in particular, circuit condition "1" results (by changing the input state from "10" to "00"), we must next concentrate our attention on the first row of the flow table, because the circled "1" entry appears in this row.

The switching circuit which we have used in this section on analysis is one which is designed to be used in conjunction with an electrically operated lock. The single output of the unit is to act as the switch which energizes the lock so that it can be opened. The two input terminals are connected to two keys, K_1 and K_2, respectively—each of which has a single normally open contact. The "combination" which is to open the lock is the following (the numbers refer to entries in Table VI):

Starting with both keys released (1), K_1 is first depressed (3) and then released (9); K_2 is then depressed (10) and released (4); finally, K_1 is again depressed (2) and the lock is to open (8) when K_1 is then released.

After the lock has been opened (8), depressing either or both of the keys (5, 6, or 7) will allow it to be locked again. If a mistake is made in working the "combination" for the lock, then one of the circuit conditions 5, 6, or 7 will result, and the combination may be started again only after both keys are first released (1).

Summary of Material on Analysis

The transition index τ_R has been defined. Its use led to the composite transition matrix τ. τ and Z (the output matrix) give us all the information that the original circuit diagram can. With a knowledge of the succession of input states, we may determine—with the aid of τ—the states of each relay of the switching circuit. Then, by looking at the proper entry in Z, the output state is determined.

We have concerned ourselves only with the relative order of state changes, and not with the actual timing in the circuit. With this outlook, we have seen some of the situations which may arise. For a circuit with no sustained cycles [9] or potentially critical race conditions, we have defined a flow table F which is useful in determining ultimate circuit conditions, if the input state is changed only after the secondary relays have reached stable states.

III. THE SYNTHESIS OF A SEQUENTIAL RELAY CIRCUIT

Introductory Remarks

In this section our attention will be limited to those design problems in which the input state changes only after a stationary output state has been reached, and in which such an output state will always occur: in other words, those circuit actions which could be those of a relay circuit with ultimately stable terminal action.

The most difficult and the most important part of a synthesis of a relay switching circuit—as with most other syntheses—is the problem of saying what we want to do. The flow table does for sequential relay circuits which have ultimately stable output what the table of combinations does for combinational-type contact networks. In each case the table gives in a precise way meanings that dozens of qualifying statements in a word specification might never be able to convey.

We must keep in mind the situation which faces the designer. First, he would like to specify what the circuit output state is to be for all possible sequences of input states, and not for just a single sequence. It is all very well to say what the output states should be for some "normal" sequence of inputs; but if there is even a possibility that other sequences might occur, then circuit action must be specified for these sequences also. A complete problem specification must indicate clearly what happens for each conceivable set of circumstances.

Secondly, if the circuit designer can honestly say that he cares only about the output state as some function of the input state, he cannot fairly say in the problem statement what secondary relays he wants to use. The problem statement can then be made only in terms of what signals are available (the input states) and what signals are desired (the output states).

[9] It is also possible to write flow tables for circuits with cycles or for circuits in which it is important to specify output states in addition to those which correspond to total relay states for which the circuit is stable. In such flow tables, entries of a type "ⓘj" may be utilized. An entry "ⓘj" is the "destination" of all entries with uncircled i's and it directs our attention to the entry with a circled "j" as the next circuit condition of interest.

The Problem Statement and the Flow Table

In order to illustrate the various problems which arise during the synthesis of a sequential circuit, we shall first limit ourselves to a specific example. The circuit we wish to design has some of the properties of a "delay line." It is to have two input terminals and two output terminals. We want the output state to be the same as the last previous input state. In order to illustrate how restrictions on the input state are taken care of, we shall specify that we shall use our circuit only under those circumstances which allow the input state to change by one variable at a time. In other words, for example, modification of the input state from "01" to "00" or to "11" will be allowed, but from "01" to "10" will be impossible. And so, after our synthesis is complete, it would not be fair to complain of improper circuit action for these forbidden changes of input state; for we specify here that they cannot occur.

With the restrictions named above, there will be just two separate circuit conditions possible for each input state, and each of these will be associated with a different output state. For instance, if the input state is "00," it could have been preceded by either "01" or "10." The corresponding output state then will be either "01" or "10," respectively.

TABLE VII.—*A Flow Table and Output Data for a "Delay Line" Circuit.*

(*a*) Flow Table				(*b*) Output Data	
x:				Stable Circuit Condition	Output State
00	01	10	11		
①	3	5		1	01
②	3	5		2	10
1	③		7	3	00
1	④		7	4	11
2		⑤	8	5	00
2		⑥	8	6	11
	4	6	⑦	7	01
	4	6	⑧	8	10

Consequently, there will be eight stable circuit conditions—two for each of the four possible input states. These are numbered serially and listed as circled entries in Column (*a*) of Table VII, each in a separate row and each in the proper column. The output state associated with each is listed in Column (*b*).

Next we list the uncircled entries of the flow table, making use of the three rules stated previously (see p. 173), and being careful to make certain the conditions in the word statement of the problem are satisfied.

The positions in the flow table which correspond to unallowable input transitions are left blank.[10]

The first row of our derived flow table informs us, for instance, that circuit condition "1" (which may occur for input state "00") may lead to either circuit condition "3" or "5," and Table VII, Column (*b*) verifies that each of these latter conditions will give the proper output state, "00."

Condensation of the Flow Table by Row Merging

From our discussion of flow tables in the section on analysis, we know that there is going to be one secondary relay state assigned to each row of the flow table. If our object is to reduce the number of secondary relays as far as possible, we would like to be sure that our

Table VIII.—*Hypothetical Derivation of a τ Matrix Corresponding to Table VII.*

(*a*) Flow Table

y: \ x:	00	01	10	11
00	①	③	5	7
11	②	3	⑤	8
10	1	④	6	⑦
01	2	4	⑥	⑧

(*b*) The τ Matrix

y: \ x:	00	01	10	11
00	00	00	11	10
11	00	11	00	10
10	10	00	11	00
01	10	11	00	00

problem has been stated in a flow table with as small a number of rows as possible. Sometimes it is possible to *merge* two rows of a flow table. Each such *merger* will reduce the number of rows by one. The rule for merging of rows in a flow table is this:

Two or more rows of a flow table may be merged if—and only if—for each input state, these rows do not have conflicting entries. An entry which appears in any one of the merged rows will appear in the composite row. Those entries which are circled in any one of the merged rows will be circled in the row resulting from the merger.

Application of this rule to Column (*a*) of Table VII indicates that the following pairs of rows may be merged: one and three, two and five, four and seven, and six and eight. The condensed flow table which results from making all four mergers is given in Table VIII.[11] The

[10] We may also leave a position in a flow table vacant if, for the corresponding input-state transition, we do not care to define the circuit action to follow. See the further discussion of this point in *Input-Output Sets*, page 183 of this paper.

[11] In our example the result of making all possible row mergers gives an answer which is independent of the order in which the mergers are made. In *Requirements for a Unique Flow-Table Condensation*, page 188 of this paper, however, it will be proved that the result of condensing a flow table to the point where no further mergers are possible may not be unique if (as in our example) there are "vacancies" in the original table: that is, if some modifications of input state are prohibited.

merger of rows one and three, for example, results in the first row of this new table.

The first row of Column (*a*) in Table VIII preserves the information contained in each of the component rows. It retains the information that circuit condition "1" leads to condition "3" or condition "5," depending on whether the input state is changed from "00" to "01" or to "10," respectively. Similarly, this first row tells us (as did the third row of Table VII, Column (*a*)) that circuit condition "3" may lead either to condition "1" or to condition "7," depending on the change of input state.

In the circuit which we shall synthesize from the condensed table, circuit condition "3" would lead to circuit condition "5" *if* the input state could be changed from "01" to "10." But the "if" is an enormous one; for we stated firmly during the specification of the problem that this transition in input state could not occur. Thus it is important to recognize that, since this information which the top row of Table VIII, Column (*a*) gives is about a hypothetical change in input state, it will neither be of value to us nor cause us trouble. Similar situations are apparent in each of the four rows in this same Column (*a*).

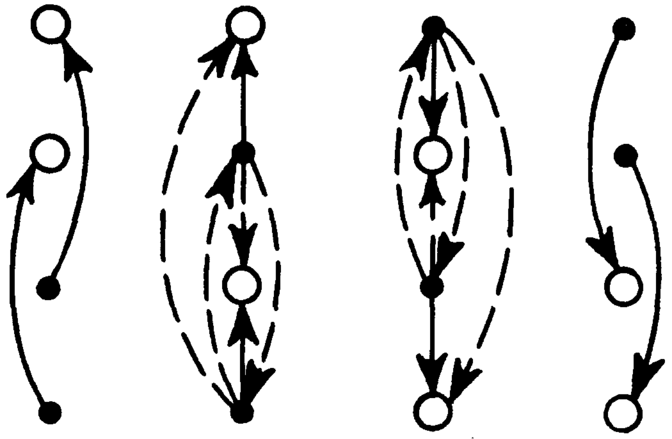


FIG. 6. Transition diagram for the derivation in Table VIII.

The Assigning of Secondary Relay States to the Rows of the Flow Table

Since there are four rows in the condensed flow table, it will be necessary to have at least two secondary relays in order to assign a distinct secondary relay state to each row. If we were to hypothesize that the operating and release times of each secondary relay were precisely the same, and that this time was the same for each of the secondary relays, then the problem of assigning secondary relay states to the rows of the flow table would be easy indeed. For then we could assign them in some arbitrary fashion to the rows, as has been done in Column (*a*) of Table VIII. The assumption that all operating and release times were all the same would guarantee that all race conditions would end in "ties," and we could then form the composite transition matrix of Column (*b*) of Table VIII, which corresponds to the diagram of Fig. 6. The transitions which correspond to the "tied races" are indicated by the solid lines in the second and third columns of the figure.

From a practical point of view, the presence of race conditions such as those discussed above is perilous; for if the operating characteristics of a relay change slightly over a period of time, one or more of the transitions indicated by dotted lines becomes possible. These transitions would result in improper operation of the switching circuit.

We can avoid the difficulties of race conditions in which "ties" *must* be achieved in order for proper circuit operation to occur (and therefore in which some entries in τ *must* indicate two or more relays to be unstable). What we need to do is to assign secondary relay states to the flow table in such a way that in τ, only *one* secondary relay is indicated to be unstable at a given time. After having done this, we may then reconsider some of the entries in τ and allow, if we desire, *noncritical* race conditions to exist.

For our present example, one assignment of secondary relay states which avoids critical race conditions is that given in Table IX, Column (*a*).

TABLE IX.—*An Alternate Development of τ and Y from Column* (*a*) *of Table VII.*

(*a*) Flow Table

y: \ x:	00	01	10	11
00	①	③	5	7
01	②	3	⑤	8
10	1	④	6	⑦
11	2	4	⑥	⑧

(*b*) τ Matrix

y: \ x:	00	01	10	11
00	00	00	01	10
01	00	01	00	10
10	10	00	01	00
11	10	01	00	00

(*c*) Y Matrix

y: \ x:	00	01	10	11
00	00	00	01	10
01	01	00	01	11
10	00	10	11	10
11	01	10	11	11

Derivation of the τ and Y Matrices

The τ matrix corresponding to our choice of secondary relay states is given in Table IX, Column (*b*). By reversing the procedure of analysis, we add cyclicly to each entry in τ the secondary relay state assigned to the row in which the entry is found. The Y matrix of Column (*c*) results.

Derivation of the Z Matrix

By consulting the output data of Table VII, Column (*b*), we may immediately assign to the Z matrix those output states which correspond to the eight stable circuit conditions of Table IX, Column (*a*). The result is the partially completed matrix of Table X.

TABLE X.—*Illustrating Development of the Z Matrix.*

(*a*) Partial *Z* Matrix

y \ x:	00	01	10	11
00	01	00		
01	10		00	
10		11		01
11			11	10

(*b*) Completed *Z* Matrix

y \ x:	00	01	10	11
00	01	00	00	01
01	10	00	00	10
10	01	11	11	01
11	10	11	11	10

If we can honestly say that we are concerned with the output state *only* after all the secondary relays of the switching circuit become stable, then any choice whatsoever may be made in the output states which are used to complete the *Z* matrix.

However, there are certain advantages to completing the matrix as we have indicated in Table X, Column (*b*). Here we have assigned to each entry in *Z* the output state associated with the corresponding entry in *F*, even if the entry is uncircled in *F*. For example, both the (01;00) and (01;01) positions in *Z* are given the output state "00," since these correspond to those positions in *F* which contain "3" as an entry, and since the circuit condition "3" is associated with the output state "00." If *Y* is completed in this way, we assure that, as the input state is changed from its present value to a new value, the output state immediately becomes and remains constant at the value corresponding to the ultimate circuit condition.

TABLE XI.—*The Derived Tables of Combinations.*

x_1	x_2	y_1	y_2	Y_1	Y_2	Z_1	Z_2
0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	0
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	0	1	0	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	0	0	1
1	1	1	1	1	1	1	0

The Tables of Combinations and the Final Circuit

We may reverse the steps taken in the section on analysis and decompose the Y and Z matrices—listing the entries in tables of combinations instead of in the matrix form. The data in Table XI result from the breaking down of the entries found in Tables IX(c) and X(b).

The purpose of this paper is to demonstrate that the problem of the design of sequential relay circuits may be reduced to the problem of the design of combinational-type contact networks. Our task is then complete, and the designer of combinational circuits may now take the data in Table XI and design corresponding secondary relay controlling networks and the output networks. One possible realization of these is given in the final circuit diagram of Fig. 7.

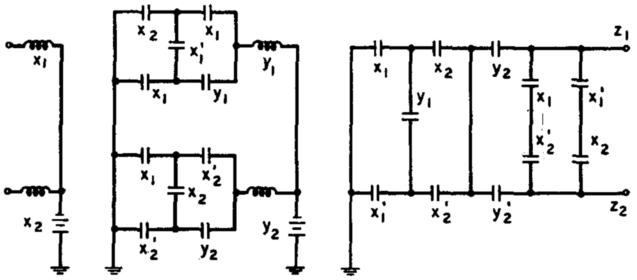


FIG. 7. The synthesized "delay-line" circuit.

IV. FLOW TABLE MANIPULATION

Simplification of the Original Problem Statement

In our zest for listing all possible circuit actions in a problem specification, it is conceivable that we may say more than we actually need to.

TABLE XII.—*The Simplification of a Problem Statement.*

(a)					(b)					(c)				
x:					x:					x:				
00	01	10	11	Output State	00	01	10	11	Output State	00	01	10	11	Output State
①	7	2	6	0	①	7	2	6	0	①	7	2	6	0
1	4	②	3	0	1	4	②	3	0	1	4	②	3	0
1	4	5	③	1	1	4	5	③	1	1	4	5	③	1
1	④	5	6	0	1	④	5	6	0	1	④	5	6	0
1	4	⑤	6	0	1	4	⑤	6	0	1	4	⑤	6	0
1	4	5	⑥	0	1	4	5	⑥	0	1	4	5	⑥	0
1	⑦	5	8	0	1	⑦	5	3	0	1	⑦	5	3	0
1	4	5	⑧	1	1	4	5	③	1					

Two or more circuit conditions which we have listed separately in a flow table might, if proper means of analysis were available, be recognized as a single condition. The elimination of this redundancy in the problem statement may be quite important to us if our object is to get as simple a circuit realization as possible. In the following sections we show that this overstatement of a problem can occur, and we shall give rules for detecting it.

As an example to show how redundancy in a flow table might arise, let us examine Table XII, Column (*a*) and the corresponding output data. This table specifies a circuit which delivers an output ground only at the end of the input sequence "00" → "10" → "11" or at the end of the input sequence "00" → "01" → "11." The first sequence corresponds to the sequence of circuit conditions "1" → "2" → "3," and the second sequence to the sequence of circuit conditions "1" → "7" → "8." The word specification of the problem is satisfied in the flow table, but we may suspect that "3" and "8" could actually be considered as the same circuit condition, since each of them is associated with the same input state and the same output state.

Let us assume, tentatively, that conditions "3" and "8" are equivalent. If this is true, any possible sequence of input states must give identical outputs whether we start with the circled entry "3" or with the circled entry "8." The third and eighth rows of Column (*a*) of Table XII tell us that given changes of input state result in the same ultimate circuit conditions, whether we start initially from "3" or from "8"; and so our original assumption is valid.

Recognition of the equivalence of conditions "3" and "8" allows us to reduce the number of circled entries in the flow table by replacing the designator "8" with the designator "3." If this is done everywhere in the table, Column (*b*) is the result. In this column the eighth row is now redundant and it may be eliminated, as in Table XII, Column (*c*). The effect of this procedure can be expressed alternatively by saying that we have merged rows three and eight, and that the merger was valid because circuit conditions "3" and "8" were equivalent.

Of course we may continue to condense the flow table by merging the fourth, fifth, and sixth rows into a single row. These latter mergers, however, would not change the number of circled entries in the table.

Now that our appetite for simplifying flow tables has been whetted, we may be tempted to see equivalence when it does not exist. In Table XII, Column (*c*), for instance, circled entries "4" and "7" are both associated with the same input state and both give the same output state. If the initial circuit condition is "4," a change of input state from "01" to "11" produces the condition "6." On the other hand, if the initial circuit condition is "7," the same change of input state (from "01" to "11") gives the condition "3." But conditions "3" and "6" cannot be equivalent because they give different output states. Therefore "4" and "7" cannot be considered equivalent either.

Table XIII.—*The Simplification of a Problem Statement.*

(*a*)

x:				Output
00	01	10	11	State
6	①	4	7	10
6	②	3	7	10
5	1	③	8	11
5	2	④	8	11
⑤	2	3	7	11
⑥	1	3	8	00
5	1	4	⑦	00
6	2	4	⑧	01

(*b*)

x:				Output
00	01	10	11	State
6	①	3	7	10
6	①	3	7	10
5	1	③	8	11
5	1	③	8	11
⑤	1	3	7	11
⑥	1	3	8	00
5	1	3	⑦	00
6	1	3	⑧	01

(*c*)

x:				Output
00	01	10	11	State
6	①	3	7	10
5	1	③	8	11
⑤	1	3	7	11
⑥	1	3	8	00
5	1	3	⑦	00
6	1	3	⑧	01

Sometimes the reasoning is not so straightforward as it was in the above example. For instance, in Table XIII, Column (*a*), we might examine for equivalence the entries "1" and "2." If we assume for the time being that they are equivalent, then examination of the flow table indicates that the validity of our assumption depends on whether or not entries "3" and "4" are equivalent. When we look at the third and fourth rows, we find in turn that "3" and "4" are equivalent if "1" and "2" are. But this follows from the original assumption. Similar reasoning would hold if we were to assume initially the equivalence of "3" and "4." Our conclusions must be: "1" and "2" are equivalent, and so are "3" and "4."

Because of the equivalences discovered above, we may condense Column (*a*) of Table XIII by merging the first and second rows and by merging the third and fourth rows. The procedure is first to replace "2" by "1" and "4" by "3" everywhere in the table (see Column (*b*)) and then to eliminate the redundant rows as in Column (*c*).

Statement of Rules

With the aid of reasoning which we have done above, we now define equivalence as follows:

Two circled entries of a flow table may be considered equivalent if, and only if,

1. each entry appears in the same column of the flow table (is associated with the same input state), and if
2. each entry is associated with the same output state, and if,
3. for each sequence of input states which could start equally well from either of the two circuit conditions involved, the corresponding output sequence is independent of which circuit condition was used as the starting point.

Another way of saying this same thing is this: In a flow table, two circled entries which are associated with the same input state and with the same output state are equivalent *unless* there exists some sequence of input states which may start from either of these two circled entries, and yet which gives corresponding output sequences which differ from each other.

There will be further illustration of these qualifications in the following sections.

Problem Statements Which Have a Unique Simplification

Input-Output Sets

The circled entries of a flow table may be partitioned into mutually exclusive *input-output sets.* Within each such set the members all have a given input state and a given output state in common. If two entries belong to different input-output sets, then either their input states or their output states, or both, differ. For example, in Table XIV, Column (*a*) we may place the entries into the following input-output sets:

(1,7,9,12); (5,14); (2,6,11); (4,8,13); and (3,10,15).

Notice that the pattern of vacancies in the rows corresponding to the members of each set is the same. For instance, the second, sixth, and eleventh rows have the following patterns of entries:

5	②		3
5	⑥		10
5	⑪		3

In each of these rows the position in the third column is left vacant.

For flow tables which are constructed in such a way that the vacancy pattern in the rows corresponding to the members of any given input-output set is the same for all members of the set, there is a unique way of eliminating redundancy and therefore a unique minimum-circled-entry form of the table. This statement will be proved later in the section entitled *Equivalence-Sets* (see page 185).

Meanwhile, let us note that a vacancy in a row of a flow table may indicate one of two things: Either (1) the input transition corresponding to the vacancy is impossible, or (2) for this transition we do not care to define the resulting circuit action. In the latter case we must be satisfied with whatever circuit action actually occurs when the relay circuit is in operation.

When we have a flow table before us, we need not be concerned with the reasons for vacancies—only that they exist. Therefore, for purposes of the following discussion, let us assume that vacancies in a flow table indicate that the corresponding transition in input state is impossible.

Testing for Equivalence

Imagine now that we have before us two identical relay switching circuits of the kind described by a given flow table. Assume also that this flow table meets the requirement that the same vacancy pattern shall exist for all members of a given input-output set. If we wanted to test two different circuit conditions, "i" and "j," for equivalence, we might do this by putting the two switching circuits into the conditions "i" and "j," respectively.

Imagine now that a given sequence of input states is impressed upon each of the circuits. If, at any time in this sequence, we find that an input transition which is possible for one circuit is not possible for the other, we might immediately conclude that the respective conditions of the two circuits are not listed in the same input-output set. Therefore we would have found one sequence of input states which, starting with the initial circuit conditions "i" and "j," did not result in the same output sequences. We should then conclude that "i" and "j" are not equivalent.

TABLE XIV.—*A Unique Simplification of a Problem Statement.*

(*a*)

x:				
00	01	10	11	Output State
①	11	4	10	01
5	②		3	11
5	2	13	③	11
12		④	15	00
⑤		8		10
14	⑥		10	11
⑦	6	8	3	01
7		⑧	3	00
⑨	11	13	10	01
12	6	13	⑩	11
5	⑪		3	11
⑫	2	4	15	01
1		⑬	10	00
⑭		8		10
1	6	4	⑮	11

(*b*)

x:				
00	01	10	11	Output State
①	2	4	10	01
5	②		3	11
5	2	4	③	11
1		④	10	00
⑤		8		10
5	⑥		10	11
⑦	6	8	3	01
7		⑧	3	00
①	2	4	10	01
1	6	4	⑩	11
5	②		3	11
①	2	4	10	01
1		④	10	00
⑤		8		10
1	6	4	⑩	11

(*c*)

x:				
00	01	10	11	Output State
①	2	4	10	01
5	②		3	11
5	2	4	③	11
1		④	10	00
⑤		8		10
5	⑥		10	11
⑦	6	8	3	01
7		⑧	3	00
1	6	4	⑩	11

With the aid of the reasoning above, we know that if any two circuit conditions are equivalent to a third, then any input sequence which is possible starting from any one of these conditions is possible starting from the other two, and corresponding output sequences are the same for all. It follows that each of the three conditions is equivalent to the others.

Equivalence-Sets

By the logical extension of the argument above, we know that, for a flow table meeting our restriction of uniform vacancy pattern for all members of an input-output set, the input-output sets themselves may be partitioned into *equivalence-sets.* These equivalence-sets are to be defined in such a way that each member of the set is equivalent to every other member of the set, and so that no member of one equivalence-set is a member of another.

For Column (*a*) of Table XIV, we may—by techniques discussed in preceding sections—determine that the equivalence-sets are: (1,9,12); (7); (5,14); (2,11); (6); (4,13); (8); (3); and (10,15).

It is easiest to test for equivalence when a flow table is in its *primitive* form: that is, with one circled entry per row. If we have determined that the flow table meets our required restrictions on the placement of vacancies, we may place the entries in equivalence-sets. If our object is to simplify the problem statement as far as possible (reduce the number of circled entries to a minimum), the best we can do is to replace each member of an equivalence-set by a single entry from that set, and then to eliminate the resulting superfluous rows of the table. This minimum number of circled entries is clearly unique, and is the same as the number of equivalence-sets.

In Column (*b*) of Table XIV, we have replaced the members of each equivalence-set by the member of this set which has the lowest numerical designation, and then we have eliminated the redundant rows. The table we have derived (Column (*c*)) is the simplest possible way of stating the terminal action specified by the original flow table of Column (*a*).

A Problem Simplification Which is not Unique

If a circuit is described by a primitive flow table which does not meet the requirements of uniformity of vacancy pattern within each input-output set, then it may not be possible to partition the entries of the flow table into equivalence-sets. Practical situations in which there may not be this required uniformity may occur if the input is the output of another sequential switching circuit, or if a human being arbitrarily restricts the input transitions allowable in accordance with his knowledge of the internal state of the circuit.

Table XV.—*Two Simplifications of the Same Problem Statement.*

(*a*)

x:				Output
00	01	10	11	State
①	3		5	01
②	7		8	01
6	③	4	5	00
1		④		10
9		4	⑤	11
⑥	3	4		01
9	⑦		8	11
2	3		⑧	01
⑨		4	8	01

(*b*)

x:				Output
00	01	10	11	State
①	3	4	5	01
②	7	4	8	01
1	③	4	5	00
1		④		10
2		4	⑤	11
2	⑦		8	11
2	3		⑧	01

(*c*)

x:				Output
00	01	10	11	State
①	3		5	01
②	7		8	01
6	③	4	5	00
1		④		10
6		4	⑤	11
⑥	3	4	8	01
6	⑦		8	11
2	3		⑧	01

In Column (*a*) of Table XV, the flow table does not meet the restrictions of uniformity of vacancy pattern; see, for example, the first and sixth rows. Inspection of the flow table and the output data in Column (*a*) shows us that the only input sequence which may start with both condition "1" and condition "6" is that beginning "00, 01, · · ·," and that for this sequence, condition "3" follows immediately whether we start with "1" or with "6." Thus "1" and "6" have a sort of equivalence. We shall call it *pseudo-equivalence* in order to distinguish it from the true equivalence, which we discussed earlier (see section beginning on page 183).

By reasoning similar to that used above, we may establish that entries "2" and "9" possess this pseudo-equivalence also. And the same is true for entries "6" and "9." The pseudo-equivalences which we have found (there are no more equivalences of any kind) in the flow table of Column (*a*) are summarized here.

$$\text{“1”} \leftrightarrow \text{“6”} \qquad \text{“2”} \leftrightarrow \text{“9”} \qquad \text{“6”} \leftrightarrow \text{“9”}$$

Notice that we cannot form mutually exclusive equivalence-sets as we did with the tables meeting the uniform vacancy requirement.

The inability to form sets of equivalent entries leads to situations which were not possible for flow tables meeting the uniform vacancy restriction. If we replace "6" by "1" and "9" by "2" in Column (*a*) of Table XV, and if we then merge the pairs of rows which contained these circled entries, the flow table of Column (*b*) can be derived. (When we compare, for example, the first row of Column (*b*) with the first and sixth rows of Column (*a*), we find that we have retained the information in the two component rows.) If, instead, we replace "9" by "6" everywhere in Column (*a*), and then merge the rows containing these circled entries, Column (*c*) follows.

Now consider Columns (*b*) and (*c*) of Table XV, both of which represent the same problem, and in neither of which can the number of circled entries be reduced further. It might well be that two different people would have stated the problem in these two forms in the first place, and so we cannot guarantee that there is a unique "simplest" primitive form of the flow table for the general sequential circuit problem.

Row Mergers

An Example in Which the Flow-Table Condensation is not Unique

Column (*a*) of Table VIII had a unique minimum-row form in which no further row mergers were possible. In general, however, when a flow table has "vacancies" there may not be a unique minimum-row form. In the flow table of Column (*a*) of Table XVI, for example,

TABLE XVI.—*Illustrating Non-Unique Flow-Table Condensations.*

(*a*)

x: 00	01	10	11
①	2	5	
1	②		4
6		③	8
	2	3	④
1		⑤	4
⑥	2	3	
1	⑦		8
	7	3	⑧

(*b*)

x: 00	01	10	11
①	②	⑤	4
6	7	③	⑧
⑥	2	3	④
1	⑦		8

(*c*)

x: 00	01	10	11
①	2	⑤	4
1	②	3	④
⑥	2	③	8
1	⑦	3	⑧

let us assume that the output states associated with the various circuit conditions are such that no redundancies exist, and therefore that no two circled entries are equivalent. In this table, any one of the following pairs of rows may be merged:

1, 2　　1, 5　　2, 4　　2, 5　　3, 6　　3, 8　　4, 6　　7, 8

If we merge rows one, two, and five; rows three and eight; and rows four and six; the result is that shown in Column (*b*). If, instead, we merge rows one and five, rows two and four, rows three and six, and rows seven and eight, the resulting condensed flow table is that of Column (*c*).

The flow tables of Columns (*b*) and (*c*) in Table XVI do not have rows which can be merged further; yet each contains the same information that the table of Column (*a*) does. From this simple example, therefore, we have demonstrated that, in general, a flow table cannot—by row merging—be reduced to a unique minimum-row form.

Requirements for a Unique Flow-Table Condensation

The result obtained in the previous section leads us naturally to the question: "Are there flow tables in which the operation of row merging leads us to a unique minimum-row form of the table?" The answer is "Yes." If a flow table has no "vacancies" (that is, if no input transitions are prohibited), there is such a unique form. We shall see below the reasoning behind our affirmative answer.

For any flow table of the "completely filled" type, we may form mutually exclusive sets of rows in which the pattern of entries in the rows of each set is the same. (The first three rows in Column (*a*) of Table XVII comprise such a set.) Then all the rows within each set

Table XVII.—*Illustrating a Unique Flow-Table Condensation.*

(*a*)

x: 00	01	10	11
①	3	2	8
1	3	②	8
1	③	2	8
5	7	2	④
⑤	7	2	4
1	3	⑥	4
9	⑦	6	8
9	7	6	⑧
⑨	7	6	8

(*b*)

x: 00	01	10	11
①	③	②	8
⑤	7	2	④
1	3	⑥	4
⑨	⑦	6	⑧

may be merged to give a single composite row which will have the same entry pattern as the members of the set. After all the rows within each set have been merged, the resultant condensed flow table has as few rows as we can obtain. This minimum-row result is unique, and independent of the order in which the mergers were made.

By way of illustration of the principles stated above, consider Column (*a*) of Table XVII. The entry patterns present in the various rows of the table allow the following mutually exclusive sets of rows to be formed: (1,2,3); (4,5); (6); and (7,8,9). The minimum-row form of this table is found by merging all rows of a set with each other. The result in our present example is the flow table shown in Column (*b*) of Table XVII.

Row Splitting

For the sake of completeness we must include among the possible manipulations on a flow table a process which is the opposite of row merging. It will be called *row splitting.* Future investigation may show that this process is sometimes useful in modifying a flow table so that the assignment of secondary relay states is made easier.

When a row of a flow table is split into several other rows, the pattern of entry designations in each of these new rows must be equivalent to that of the original row. In the flow table of Column (*a*) of Table XVIII, one way of splitting the row containing the circled entries

TABLE XVIII.—*Illustrating Splitting of Rows in a Flow Table.*

(*a*)

x: 00	01	10	11
①	6	⑤	3
1	②	4	3
7	2	⑧	③
⑦	⑥	④	3

(*b*)

x: 00	01	10	11
①	6	5	3
1	6	⑤	3
1	②	4	3
7	2	⑧	③
⑦	6	4	3
7	⑥	④	3

one and five, and of splitting the row containing the circled entries four and six, is that shown in Column (*b*). In this derived table the rows resulting from the splitting process give the same description of circuit action that the original flow table did, and we could obtain again the original flow table by making all possible mergers in Column (*b*).

Another more complicated type of row splitting occurs if the row we attempt to split has but one circled entry. Then the effect is to increase the number of circled entries in the flow table; or we may say that the single circled entry of such a row is itself split. The rules for *splitting an entry* into two or more parts are easy to state. If we split an entry "i" into m different entries, we take the row of the flow table which contains the circled entry "i" and replace it by m rows which contain circled entries "i_1," "i_2," $\cdots$, "i_m." Each of these m circled entries is to be equivalent to the others. Thus each will be in the same column as the original entry "i," and the output state associated with each will be the same as that of the original entry. In each of the rows of the new flow table we insert uncircled entries which are either the same as the corresponding entries in the original table or equivalent to them.

By way of illustration, let us, in Column (*a*) of Table XIX, split the entry "1" into three parts, and the entry "3" into two parts. One way of doing this is that given in Column (*b*). Notice that in this latter table the uncircled entries "1" and "3" have been replaced by equivalent

Table XIX.—*Illustrating Splitting of Entries in a Flow Table.*

(*a*)

x: 00	01	10	11
$\textcircled{1}$	3		4
7	3	$\textcircled{2}$	
	$\textcircled{3}$	5	4
7	6	5	$\textcircled{4}$
1		$\textcircled{5}$	
7	$\textcircled{6}$	2	
$\textcircled{7}$	3		4

(*b*)

x: 00	01	10	11
$\textcircled{1_1}$	3_1		4
$\textcircled{1_2}$	3_2		4
$\textcircled{1_3}$	3_2		4
7	3_2	$\textcircled{2}$	
	$\textcircled{3_1}$	5	4
	$\textcircled{3_2}$	5	4
7	6	5	$\textcircled{4}$
1_2		$\textcircled{5}$	
7	$\textcircled{6}$	2	
$\textcircled{7}$	3_1		4

entries everywhere in the table. We could, of course, simplify the table of Column (*b*) to produce again the flow table of Column (*a*).

Combinations of the techniques above may be used to split the rows in a flow table. (See the example leading up to Table XXV.) The test for the validity of such splits is always to determine whether or not the rows resulting from each split may be merged again to give the original row.

(*To be continued.*)