Some Problems in Relay Circuit Design

J. A. BRZOZOWSKI, MEMBER, IEEE

Abstract

In the design of sequential relay circuits one often has to consider the circuit as a part of a larger system, and one must take into account the nature of the inputs to the circuit. Frequently, the inputs are two-terminal contact networks with one terminal grounded; such inputs will be called simple. This paper examines the formal design techniques as applied to circuits with simple inputs. It is shown that all relays should be considered as secondary, from the point of view of both races and minimization. Several other shortcomings of the present theory are pointed out. Properties of circuits with simple inputs are examined. It is shown that there are flow tables not realizable without races with simple inputs, and that the presently known secondary assignment methods are not applicable to circuits with simple inputs.

Motivation

The application of formal design techniques for sequential circuits is, in practice, limited to problems of relatively small size; for example, one does not attempt to construct a digital computer or a telephone switching system by treating it as a sequential circuit described by a single flow table. On the other hand, formal methods are certainly very helpful in designing small parts of a large system, which are then combined with the aid of more intuitive methods. It is felt that the existing methods can be extended by taking into account the environment of the circuit. In this paper an attempt is made to consider a relay circuit[1-3] as a part of a larger system. In particular, this point of view leads to certain restrictions on the nature of the inputs to the circuit.

We begin by considering an example. Suppose we are given the flow table T_1 of Fig. 1(a) which is to be realized by a relay circuit. Using the ordinary textbook[1-3] techniques, it is seen that the number of rows cannot be reduced and that one secondary relay is necessary. One secondary relay is also sufficient, because there can be no races with only one internal variable. Let $y_1=0$ for A and $y_1=1$ for B. Then the assignment of Fig. 1(b) results. The circuit functions for this assignment are:

$$Y_1 = x_1(\bar{x}_2 + y_1), \qquad z = y_1.$$

Here we encounter our first problem: In order to realize Y_1, $\bar{x}_2$ is necessary, but $\bar{x}_2$ is not in general available, if the input is a simple key or a two-terminal contact network. A well-known method[1-3] of solving this problem is to use a primary relay excited directly by x_2; any number of front and back contacts are then available, at least in a theoretical treatment. Let us proceed in this fashion, but let the relay excited by x_2 be called Y_2. We arrive at the circuit C_1 of Fig. 1(c).

Now, if we were asked to analyze the circuit C_1, we would obtain the excitation table of Fig. 1(d). We can see that the behavior of C_1 is not quite the same as that predicted by the initial design:

1) There are really *two* internal variables, i.e., *two* secondary relays, one of which happens to have a rather simple excitation function.
2) There are *races* in columns (00) and (01). These races are not critical.
3) There is a *critical race* in column (11). If the circuit is in stable state (00) and the inputs change simultaneously, as the outcome of the race the circuit may either end up in state (01) with output 0 or in state (11) with output 1.
4) As a consequence of the above, the circuit *behavior* may not be as desired.

Let us consider the foregoing observations in more detail. First, we contend that the proper point of view is to consider *all relays as secondary*. There are two basic reasons for this: The primary relays may race with each other or with secondary relays, and, from the point of view of minimization, one would like to know the *total* number of relays one has to use in the circuit. In the presently known methods[2-4] for making secondary assignments, this information is not available until the circuit is designed and analyzed. Of course one may attempt to change the nature of the inputs to avoid the introduction of "primary" relays; more will be said about this later. Secondly, let us consider the problem of races: The races in columns (00) and (01) both occur only when two inputs change simultaneously, as does the race in column (11). The usual assumption is that only one input changes at a time and so these races may be considered unimportant; yet it is felt that the present methods are not quite adequate, since the original assignment [Fig. 1(b)] indicates that no races at all will occur. Furthermore, it is possible to construct examples where a race occurs with only a single input change. For the flow table of Fig. 2(a) we first use the assignment of Fig. 2(b) with the next state functions:

$$Y_1 = \bar{x}y_2 + xy_1,$$
$$Y_2 = \bar{x}y_2 + x\bar{y}_1.$$

If we replace $\bar{x}$ by $\bar{y}_3$, where $Y_3=x$ we have the table of Fig. 2(c). It can be seen that there are several races.

Most flow tables like that of Fig. 1(a) are a result of simplifying a primitive flow table which in turn arises from a word description of a problem. For example, suppose it is desired to design a circuit with two inputs x_1 and x_2 and one output z. The output is to be 1 when $x_1=1$ except that, if both x_1 and x_2 are 1, the output is to be 0, if x_2 changed to 1 before x_1 did. When $x_1=0$, the output is to be 0. The primitive flow table T_2 for this specification is shown in Fig. 3(a). We assume that we are interested in the output only when the circuit is stable. If a double change in the inputs takes place, we assume that it is acceptable to consider either input as changing first; hence, there are don't care entries (−) as shown. Here we would like to point out that the don't cares are really restricted in nature, in general: When the final circuit is designed, it usually has many more states than does the flow table. Say the circuit is in state D and both inputs change simultaneously. It is conceivable that the circuit may end up in some stable state with $z=1$, if the don't care entry is improperly filled out. Thus the don't cares in column (00) are really not don't cares, for the designer will insist that the circuit should end up in state A or another stable state indistinguishable from A. The same remarks apply to the don't cares in columns (01) and (10). On the other hand, in column (11) the don't care entry should really be replaced by "either D or E." This points out that one must be careful in interpreting the don't care entries, and for the specification given we prefer to use the table of Fig. 3(b) (ignore z_1 and z_2), since it indicates the designer's intentions more clearly. When either table of Fig. 3 is simplified, it results in the table T_1 of Fig. 1.

Let us return now to the question of inputs in a relay circuit. Very often an input x_j consists of a complicated two-terminal contact network. We have already seen that complications arise, if a complementary input is desired for the excitation of a secondary relay. One can, of course, design a complementary network for $\bar{x}_j$; this means extra cost and again is not predicted by the present methods. Secondly, there is the question of fan-out: It may be that the network realizing x_j is to be used also for other purposes, in addition to controlling the circuit being designed. But suppose we require an excitation $Y_2=x_1y_1+y_2$ and we construct this as shown in Fig. 4. Then the

Manuscript received September 9, 1964; revised March 15, 1965. This work was supported by the Northern Electric Co., Ltd.

The author is with the Department of Electrical Engineering, University of Ottawa, Ottawa, Ontario, Canada, and is a consultant to Dept. 8160, Research and Development Labs., Northern Electric Co., Ltd., Ottawa, Ontario, Canada.

[1] D. A. Huffman, "The synthesis of sequential circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, March 1954, and vol. 257, pp. 275–303, April 1954.
[2] S. H. Caldwell, *Switching Circuits and Logical Design*. New York: Wiley, 1958.
[3] M. P. Marcus, *Switching Circuits for Engineers*. Englewood Cliffs, N. J.: Prentice-Hall, 1962.
[4] D. A. Huffman, "A study of memory requirements of sequential switching circuits," Tech. Rept 293, Research Lab. of Electronics, M.I.T., Cambridge, Mass., April 1955.

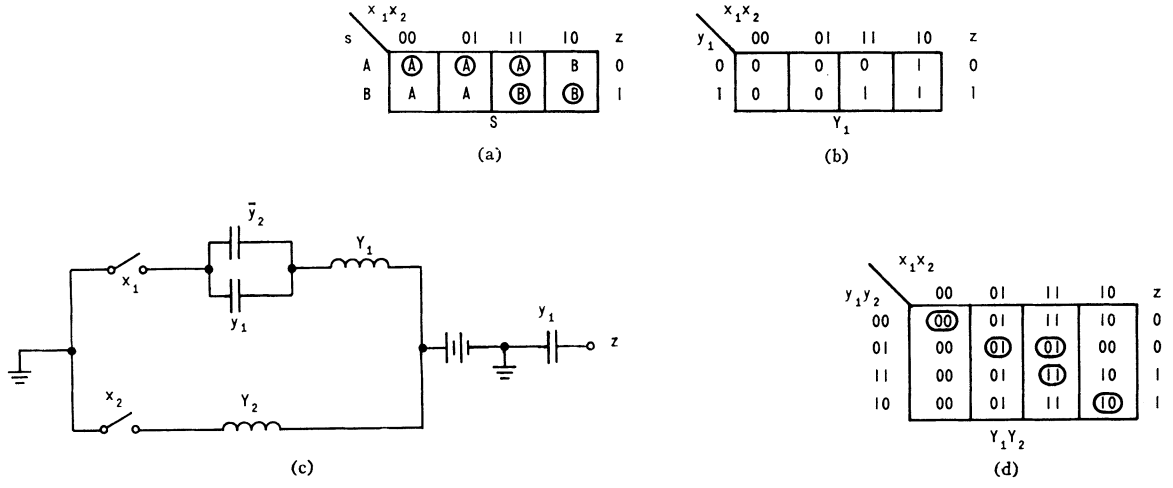

Fig. 1. (a) Flow table T_1. (b) An assignment for T_1. (c) Circuit C_1 for T_1. (d) Analysis of C_1.

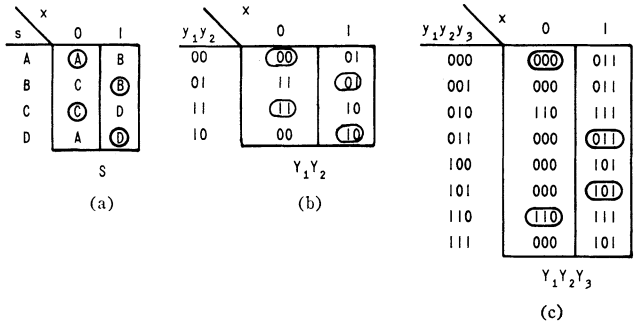

Fig. 2. (a) Flow table T_2. (b) An assignment for T_2. (c) Analysis of circuit for T_2.

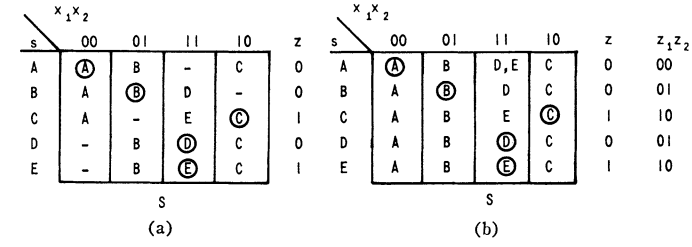

Fig. 3. (a) Flow table T_3. (b) Flow table T_4.

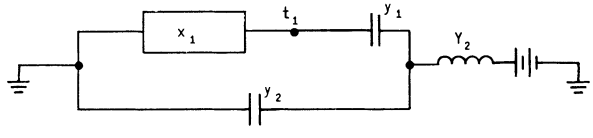

Fig. 4. Illustrating sneak paths.

function at the input terminal t_1 is no longer x_1 but $x_1+y_1y_2$. Thus a sneak path has been introduced.

In summary, there are enough difficulties with the present design techniques to motivate us to reexamine the methods.

We will assume that the flow tables which are to be realized are well-behaved in the following sense: In the primitive flow table, which is normally the first step in the design, we are interested in one step transitions between stable total states. Thus, in each row there is exactly one stable state, say (S_1, X_1) where X_1 denotes the input n-tuple and S_1 is an internal state. If the input changes to X_2, the entry in row S_1, column X_2, is S_2, and the entry in row S_2, column X_2, is also S_2. Some entries may be unspecified. This type of table is sufficient to take care of most design problems; however, we will consider other allowed forms, because the primitive flow table undergoes changes in the design process. First, more than one stable state can be found in some rows as a result of merging. Secondly, in order to avoid races we may specify multiple internal state changes which, however, always lead to a stable total state. The introduction of multiple changes may lead to a table in which some rows have no stable entry, and the unstable entries are used as intermediate steps in multiple transitions. From a problem specification we obtain a desired table T_D in which internal states are only represented by letters; in a circuit, the internal states are represented by n-tuples of binary variables, but of course each circuit has a corresponding table, T_C. In summary, we will only treat *well-behaved* tables T_D and T_C which have the property: starting in any stable state (S_1, X_1) an input change X_1 to X_2 leads eventually to a stable state (S_2, X_2). We shall assume that the intermediate internal states between S_1 and S_2 are of no consequence and any finite sequence of such intermediate steps is satisfactory. Similarly, only the outputs corresponding to stable total states will be considered important.

We shall now say that a circuit C *realizes* a desired table T_D if, and only if, for each stable total state (S_1, X_1) in T_D there is at least one stable total state (S_1', X_1) in T_C, the flow table of C, which is indistinguishable from (S_1, X_1), where the distinguishing is to be done only by outputs from stable total states.

Positive Inputs

We introduce our first restriction on the nature of the inputs by assuming that *complemented inputs are not available.* In practice the designer must decide whether this assumption is reasonable. In some cases it may be economical to redesign the input circuits to some extent in such a way that complemented inputs are available; the preceding assumption represents an extreme situation. Note that "primary" relays will be considered here as secondary and will appear naturally in an assignment as secondary relays, each of which has an excitation function equal to one of the inputs.

Lemma 1: In a circuit in which complemented inputs are not available, all excitation functions Y_j, $j=1, 2, \cdots, m$, must be positive[5] functions of the input variables $x_1, x_2, \cdots, x_n$.

[5] R. McNaughton, "Unate truth functions," *IRE Trans. on Electronic Computers*, vol. EC-10, pp. 1–6, March 1961.

The lemma is obviously true. A circuit for which the assumption holds will be said to have *positive inputs*.

Definition 1: A row R of a flow table, having $k>0$ stable next state entries (R), is said to be *precisely realized* by a circuit C, when there exists at least one state of the internal variables which is stable for all the inputs for which the next entry is R. A flow table is precisely realized by a circuit C when each of its rows having at least one stable next state entry is precisely realized by C. (Of course the transitions between states must correspond to those of the flow table.)

For example, if complemented inputs are available, the circuit corresponding to the excitation table of Fig. 1(b) precisely realizes the flow table T_1 of Fig. 1(a). On the other hand, the circuit C_1 of Fig. 1(c) does not precisely realize T_1. It is clear that T_1 is not precisely realizable by a circuit with positive inputs and one internal variable, because row A is not precisely realizable with positive inputs and one internal variable. Thus row A must be represented by more than one row of the excitation table, if the inputs are to be positive; i.e., the flow table T_1 must be expanded. This can be done as shown by T_5 in Fig. 5(a), where state A is represented by two states A_1 and A_2 and the entry A_1, A_2 indicates an option—either A_1 or A_2. Now it is possible to make an assignment of two internal variables as shown in Fig. 5(b), and the corresponding circuit precisely realizes T_5, whose behavior is indistinguishable from that of T_1. The excitation functions are:

$$Y_1 = y_1y_2 + x_1(y_1 + y_2),$$
$$Y_2 = \bar{y}_1 + x_2y_2,$$

and the circuit is shown in Fig. 5(c). This assignment uses only positive inputs and is completely race-free.

In general, if a table T is realizable by a circuit with positive inputs then some equivalent version T' of T is precisely realizable. For two variables there are only six positive functions as shown in Fig. 6. Since every relay excitation function must be positive in the inputs, every internal variable for a circuit with two positive inputs must appear in every row as one of the patterns from Fig. 6(a). Because this must be true of all internal variables, combinations of these patterns determine the appearance of a row. For example to precisely realize a row R [Fig. 6(b)] with two stable entries R and two unstable entries U_1 and U_2 we have the choice of any one of the rows shown in Fig. 6(b) for an assignment with two internal variables. Note that for patterns 3 and 4 of Fig. 6(b) there is a race when both inputs change from 0 to 1.

For the entire table one must consider all rows in this manner and the preceding ideas can be used as a guide; however, the problem appears to be very difficult. The assumption of positive inputs alone is, in a way, unrealistic, because *duplication* of inputs may still be required, and, if one is willing to redesign the input networks to allow duplication, then one is probably just as willing to redesign them to allow input complementation. Thus we are led to consider further restrictions.

Further Input Restrictions

We now introduce what appears to be the most severe restriction on the nature of the inputs that one is likely to need.

Definition 2: A two-terminal contact network N_i with terminals g_i and t_i, with g_i grounded, is called a *simple input*. The transmission from ground to t_i is the input variable x_i. For each x_i only one N_i is available.

Definition 3: A simple input x_i is *preserved* by a circuit, if the transmission from ground to t_i remains equal to x_i after the input network is used in the circuit. In other words, no sneak paths are introduced by the circuit.

For example, the circuit of Fig. 5(c) has simple inputs, but does not have the input preserving property for x_2, because the transmission to ground at terminal t_2 is $x_2+\bar{y}_1y_2$. This can be remedied by writing $Y_2=x_2y_2+\bar{y}_1=x_2y_2y_1+\bar{y}_1$ and adding the contact y_1 in series with y_2. In the new circuit, since $y_2y_1\bar{y}_1=0$, there are no sneak paths. However, there is still a sneak path to x_1.

Theorem 1: A circuit can be constructed with preserved simple inputs if, and only if, each relay excitation Y_k, $k=a, b, \cdots, m$ can be put in the form:

$$Y_k = k_0 + k_1x_1 + k_2x_2 + \cdots + k_nx_n, \tag{1}$$

where the k_i are independent of the inputs and

$$k_ik_j = 0 \qquad \text{for all } i, j,\ i \neq j. \tag{2}$$

Proof: If every Y_k can be written in the preceding form, then clearly we can construct the circuit as shown in Fig. 7. Each input is used only once, no complements are required, and there are no sneak paths. Hence the circuit has preserved simple inputs. Conversely, if the circuit can be constructed with preserved simple inputs, then each Y_k must be positive in each x_i. Furthermore, since one side of each input must remain grounded, it is impossible to generate terms involving products of the x_i. Hence each Y_k can be written in the form (1). Now assume that for some Y_k there is no form (1) satisfying condition (2). In any circuit realizing Y_k there must be at least one path corresponding to x_ik_i. Thus if $k_ik_j\neq0$, there is a sneak path to x_i when $k_i=k_j=x_j=1$. By such an argument we see that (2) must be satisfied for a circuit with preserved simple inputs.

The input preservation requirement limits severely the nature of the secondary assignment. If we fix the present state, then all the k_i are fixed in (1). Hence, in any row, Y_k as a function of the inputs can either be constant (0 or 1) or $Y_k=x_i$. All other possibilities are excluded. For example, in the table of Fig. 6(b) patterns 3 and 4 are no longer allowed, since they involve x_1x_2 for one of the Y_k.

Corollary: If in each row of a flow table each Y_k is either constant, or equal to one of the inputs, then a circuit with preserved simple inputs can be constructed.

Proof: Let f_0 be the sum of all the products (of $y_1, y_2, \cdots, y_m$) for which $Y_k=1$. Similarly, let f_i be the sum of products for which $Y_k=x_i$. Then

$$Y_k = f_0 + f_1x_1 + f_2x_2 + \cdots + f_nx_n,$$

where it is clear that $f_if_j=0$ for all $i, j, i\neq j$.

Consider again our example of Fig. 1. T_1 can be expanded as shown in Fig. 8(a). The assignment of Fig. 8(b) results in the functions:

$$Y_1 = x_2(\bar{y}_2 + y_1)$$
$$Y_2 = x_1 + y_1 = x_1\bar{y}_1 + y_1.$$

This assignment gives a circuit with preserved simple inputs. There is a race which is critical from the point of view of T_1 [Fig. 1(a)] but is acceptable, if the designer's intentions are interpreted as in Fig. 3(b).

Race-Free Circuits

Although one often accepts noncritical races and occasionally critical races (as in the example of Fig. 8), it is of interest to investigate the conditions under which a circuit is realizable without races. We assume that the circuit must have preserved simple inputs.

Definition 4: A secondary variable Y_k is said to be *active* in row R of a flow table if it is not constant in that row.

Theorem 2: If a row R, with at least one stable next-state entry, can be precisely realized without races by a circuit with simple inputs, then at most one secondary variable is active in that row.

Proof: Suppose $Y_a=x_i$ and $Y_b=x_j$ are two active secondary variables in row R. Let $Y_a=k_1$, $Y_b=k_2$ be the values of Y_a, Y_b which correspond to the stable entry. Because of the nature of the Y functions, all combinations of Y_a and Y_b will appear in row R; in particular, the combination $Y_a=\bar{k}_1$, $Y_b=\bar{k}_2$ will appear. This, however, represents a race; thus, at most, one active variable is allowed.

Definition 5: The *stability function* $s_R(x_1, x_2, \cdots, x_n)$ of a row R is defined:

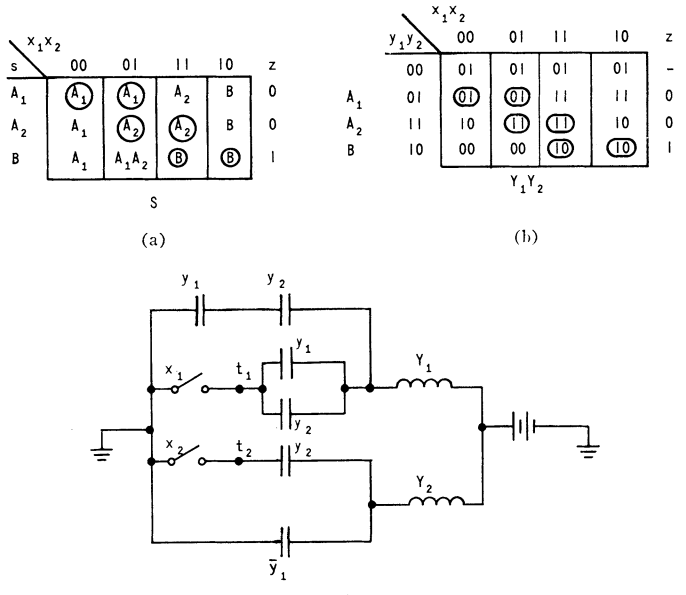

Fig. 5. (a) Flow table T_5. (b) Assignment for T_5. (c) Circuit for T_5.

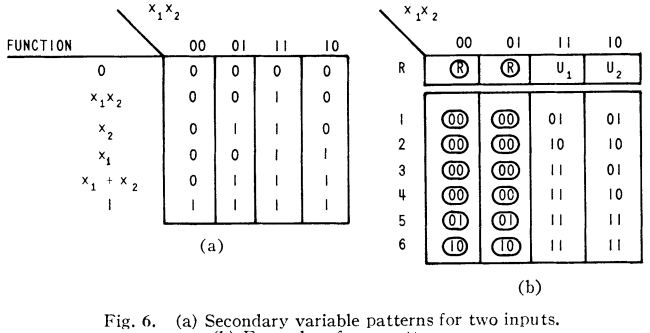

Fig. 6. (a) Secondary variable patterns for two inputs. (b) Examples of row patterns.

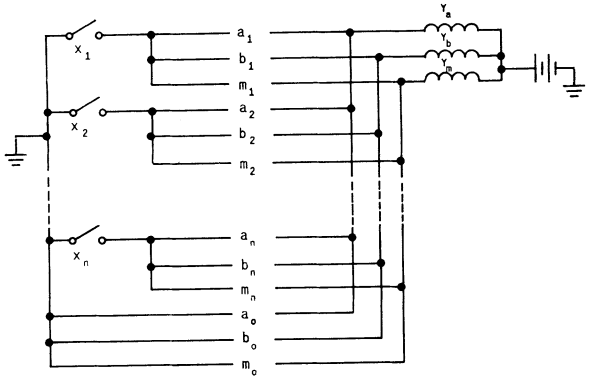

Fig. 7. General circuit with simple inputs.

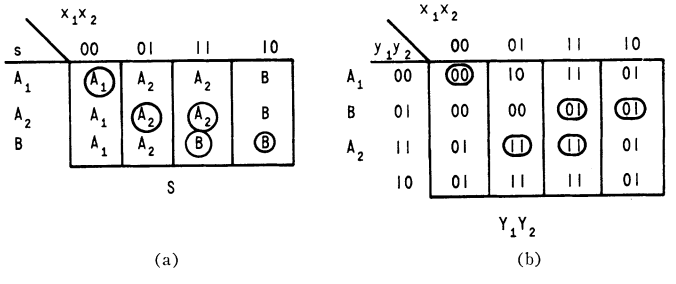

Fig. 8. (a) Flow table T_6. (b) Assignment for T_6.

$s_R(x_1, x_2, \cdots, x_n) = 1$, if the entry in the $(x_1, x_2, \cdots, x_n)$ column of row R is stable,

$s_R(x_1, x_2, \cdots, x_n) = 0$, otherwise.

Corollary 1: Under the conditions of Theorem 2, either s_R or $\bar{s}_R$ is equal to one of the inputs or is a constant.

Proof: If all entries are stable, then $s_R = 1$. If there is at least one unstable entry, then at least one active variable is required. In view of Theorem 2 there must be exactly one active variable Y. Since the inputs are preserved, $Y = x_i$ in any row. Hence either $s_R = x_i$ or $\bar{s}_R = x_i$.

Corollary 2: In any row satisfying the conditions of Theorem 2, at most two different n-tuples of secondary variables can be assigned —one for the stable entries, the other for the unstable entries.

We can now show that there are flow tables which cannot be realized with preserved simple inputs without races. Consider the flow table of Fig. 3(b), but with outputs z_1 and z_2 as shown. Now row A cannot be merged with any other row. This means that in any realization there must be a stable internal state corresponding to A under input (00) which is such that the internal variables *must change*, if the input changes. Since $s_A = \bar{x}_1\bar{x}_2$, by Corollary 1 and Theorem 2, the row is not realizable without races.

The same table with output z can be merged as in Fig. 5(a), and the assignment of Fig. 5(b) can be used. The circuit will have no races. If the excitation functions are chosen as:

$$Y_1 = y_1y_2 + x_1(\bar{y}_1y_2 + y_1\bar{y}_2),$$

$$Y_2 = \bar{y}_1 + x_2y_1y_2,$$

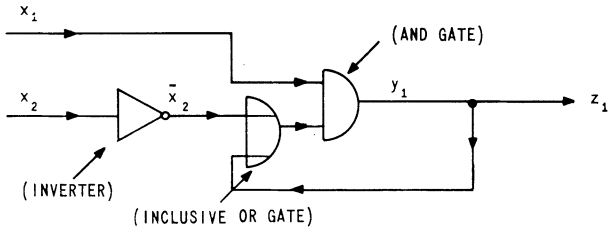

Fig. 9. Gate realization of T_1.

then the circuit has preserved simple inputs.

In general the assignment problem for race-free circuits with simple inputs appears to be difficult. The presently known[4,6] secondary assignment methods do not take into account the nature of the inputs.

Gate Circuits

In asynchronous[7] or fundamental mode[8] sequential circuits constructed from gates, the problem is somewhat different. The analogous input to a two-terminal contact network is a gate network with a single output lead whose logical variable is x_i. The first basic difference is the fact that no sneak paths exist in gate circuits. In relay circuits a simple input x_i can be used to control more than one circuit provided these circuits preserve that input. In a gate circuit the number of circuits which can be controlled by a given input depends on the fan-out capabilities of the input network. For most circuits some fan-out is available, and the usual way to increase the fan-out is to provide a (noninverting) amplifier. On the other hand, complementary inputs are usually obtained by using an inverter, which normally is an inverting amplifier.

Now it is true that, when an amplifier is inserted, some delay is introduced, and one could consider the amplifier or inverter output as an additional state variable. Note, however, that to obtain input complementation or duplication it is not necessary to introduce feedback loops or flip-flops which are normally associated with secondary variables. On the other hand, in relay circuits, complementation and duplication must be obtained by using extra "primary" relays, i.e., elements identical to secondary relays. The only distinction which could be made between the primary and the secondary is that the excitation of a primary relay is equal to one of the inputs, whereas the excitation of a secondary relay may depend on the state of its own contacts or the contacts of other relays.

Let us consider the table of Fig. 1(a) and use the assignment of Fig. 1(b) to realize the table with gates. The gate circuit is shown in Fig. 9. If we try to use analogous reasoning to that of our analysis of relay circuits, the inverter output should be considered as an added secondary variable; but, using this approach, one would have to consider the output of every gate (here also the OR gate) as a secondary variable. This is not consistent with present practice of considering only the feedback variables as state variables, and represents a more accurate but also a considerably more complicated representation of the circuit.

In the circuit of Fig. 9, if the inputs change from (00) to (11) with y_1 initially equal to 0, both inputs to the AND gate will be 1 for a short period of time and it is possible that the circuit will change to $y_1=1$, thus giving incorrect behavior. This depends on the delays through the inverter and the OR gate, and on the sensitivity of the AND gate to short pulses. If gate delays cannot be neglected, then an approach similar to that for relay circuits must be used to obtain a more accurate model of the circuit behavior.

[6] C. N. Liu, "A state variable assignment method for asynchronous sequential switching circuits," *J. Assoc. Comp. Mach.*, vol. 10, pp. 209–216, April 1963.
[7] G. A. Maley and J. Earle, *The Logic Design of Transistor Digital Computers*, Englewood Cliffs, N. J.: Prentice-Hall, 1963.
[8] E. J. McCluskey, "Logical design of digital circuits," class notes for E417, Department of Electrical Engineering, Princeton University, Princeton, N.J., 1962.

Conclusion

We have pointed out some desirable features which one would like to incorporate into formal design techniques for sequential circuits. Some general observations have been made; these are helpful in the search for nice realizations, but further work in this area is required.

Acknowledgment

The author wishes to thank R. Kenedi of Northern Electric Research and Development Labs., Ottawa, Ontario, Canada, for useful discussions which led to the formulation of some of the foregoing problems.

Comments on the Minimization of Stochastic Machines

SHIMON EVEN, MEMBER, IEEE

Introduction

In a recent note by Bacon [1], a new result about the minimization of stochastic finite state machines has been proven. Extending the theory developed by Carlyle [2], he shows that all minimal-state forms of equivalent machines are state equivalent, and they all have the same number of states. He also describes a necessary and sufficient condition for a machine to be minimal.

In the present note, we resolve several issues that have been left open, and show a minimization algorithm. The notation and terminology are explained in Bacon's note. However, it is important that the reader realizes the difference between reduced forms, in which no two states are equivalent, and minimal forms, in which no state is equivalent to a distribution of states.

On the Relation Between Minimal State Machines and Deterministic Machines in Reduced Form

In this section, we shall demonstrate that minimal state machines are a direct generalization of the reduced form for deterministic machines. Formally, the reduced forms of stochastic machines are the natural extension of the reduced form for deterministic machines, but the minimal state forms may have a smaller number of states. Bacon has shown that a minimal state form of a given machine has a unique number of states, and this adds to the importance of the minimal state forms. Therefore, it is interesting to show that the minimal form of a deterministic machine, viewed as a stochastic one, is identical to its reduced form.

Theorem: Let A be a deterministic machine and B a stochastic machine in reduced form. No state of A is equivalent to a non-trivial distribution of B.

Manuscript received July 29, 1964.
The author is with the Technion-Israel Institute of Technology, Haifa, Israel. He was formerly with Sperry Rand Research Center, Sudbury, Mass.