INFORMATION AND CONTROL 10, 292–314 (1967)

On Single-Loop Realizations of Sequential Machines[1]

J. A. BRZOZOWSKI

Department of Electrical Engineering, University of Ottawa

This paper is concerned with the problem of realizing a sequential machine M by a sequential circuit with a single feedback loop carrying a binary signal. To find such a realization, one must find a binary total-state partition, which can be used for feedback, for the flow table T' of M or for some expanded version of T. A method is presented for testing whether a given flow table possesses a feedback partition. If one exists, the method can generate all such partitions, and the corresponding single-loop realizations are easily found.

LIST OF SYMBOLS

$i, j, k, l, m, n, p, s, u, v, w, L, K, N, i_k, l_i, v_j$	Non-negative integers
M, M', M_i, M_S	Sequential machines
T	Flow table
$P(T)$	Pair table of T
$P = \{p_1, p_2, \cdots, p_u\}$	Input alphabet
$Q = \{q_1, q_2, \cdots, q_n\}$	State alphabet
$R = \{r_1, r_2, \cdots, r_w\}$	Output alphabet
p, q, r	Input, output and state variables, respectively
p'	Augmented input variable
q_λ	Initial state
r_a	Auxiliary output
Q_i	Subset of Q
Q_{ij}	Two-element subset of Q
A, B, C, D	States in examples

[1] The research reported in this paper was supported in part by the National Research Council of Canada under Grant No. A-1617 and in part by the U.S. Air Force Office of Scientific Research, Grant No. AF-AFOSR-639-65. This paper was presented at the Sixth Annual Symposium on Switching Circuit Theory and Logical Design, University of Michigan, Ann Arbor, Mich., and the majority of the results were published in the 1966 IEEE Conference Record 16C13, 84–93; October, 1965.

σ, ρ	Next-state and output functions, respectively
δ, θ, μ, ψ, ω	Combinational functions
t	Time
S, C, U	Sequential circuits
x_i, y_i, z_i	Binary input, state and output variables, respectively
y	Binary state variable
$y_{i\lambda}$	Initial value of y_i
X, Y, Z	Input, state and output-tuples
f_i, g_i	Next-state and output Boolean functions
f	Next-state Boolean function
W_i	Set of wires
w_{ij}	Signal on jth wire
W	k-tuple of wire signals
π, π_i	Total state partitions
$\prod_{i=1}^{k}$	Product
F, F_i	Feedback values
$\gamma(\pi)$	Generating function of π
$\mathbf{x}$	Input sequence
$\mathbf{s}$, $\mathbf{s}_i$	Input-output sequence
$\sigma_{\mathbf{s}}$	$\mathbf{s}$-successor function
$\times$	Cartesian product
$\vee$	Disjunction
$\in$	Is an element of
$\cap$	Intersection
$\cup$	Union
$\varnothing$	Empty set
$\bar{x}$	Complement of x

I. INTRODUCTION

The relation between the behavior of a sequential machine and the structure of the sequential circuit realizing that behavior is a current subject of investigation in sequential machine theory. One important structural property is the amount of feedback in the circuit. It has been recently shown that a sequential machine either requires no feedback or can be realized by a circuit with a single binary feedback loop. The problem of finding single-loop realizations is discussed here.

For the sake of brevity, a lot of the preliminary material is presented

here in a very brief fashion. For background on sequential machines, realizations and partitions refer to Hartmanis and Stearns (1966). Some general properties of feedback and a discussion of previous results are given in more detail by Brzozowski (1965).

II. FUNDAMENTALS

DEFINITION 1. A *sequential machine* M is a sextuple:

$M = (P, Q, R, \sigma, \rho, q_\lambda)$, where P, Q, R are finite nonempty sets:
$P = \{p_1, p_2, \cdots, p_u\}$, the *input* set,
$Q = \{q_1, q_2, \cdots, q_v\}$, the (internal) *state* set,
$R = \{r_1, r_2, \cdots, r_w\}$, the *output* set,
$q_\lambda \in Q$ is a specified *starting state*,
σ is a function from $P \times Q$ into Q (the *next-state* function),
ρ is a function from $P \times Q$ into R (the *output function*).

Consider p to be a u-valued input variable taking its value from P, and similarly define variables q and r. Then we may write:

$$q(t+1) = \sigma(q(t), p(t)), \quad \text{for} \quad t \geqq 1, \qquad q(1) = q_\lambda, \tag{1}$$

$$r(t) = \rho(q(t), p(t)), \qquad t \geqq 1. \tag{2}$$

For $t < 1$, $q(t)$ and $r(t)$ are of no interest.

Although we generally regard a sequential machine as a specification of a *behavior*, these relations immediately suggest the *structure* shown in Fig. 1, where the rectangle denotes a unit delay. The diagram can be viewed as a realization of the machine, if σ is a combinational circuit receiving a u-valued signal $p(t)$, a v-valued signal $q(t)$ and producing a

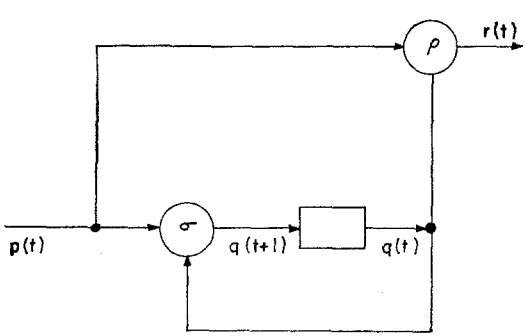


FIG. 1. Structure of a sequential machine

v-valued signal $q(t+1)$, and if a similar interpretation for ρ is used. The unit delay transmits v-valued signals.

It is clear that this is the most straight-forward multi-valued realization of a machine. It is simple because it involves only one delay and two combinational circuits. Of course, at present, there are considerable problems with realizations utilizing multi-valued logics; hence we concern ourselves mainly with binary logic. However, the general sequential circuit is defined as follows:

DEFINITION 2. A *sequential circuit* is a structure consisting of n inputs $x_1, x_2, \cdots, x_n$, m outputs $z_1, z_2, \cdots, z_m$, s unit delays with outputs $y_1, y_2, \cdots, y_s$ and $m + s$ combinational circuits, $f_1, f_2, \cdots, f_s, g_1, g_2, \cdots, g_m$ such that for $1 \leqq i \leqq s$, $1 \leqq j \leqq m$:

$$y_i(t+1) = f_i(y_1(t), y_2(t), \cdots, y_s(t); x_1(t), x_2(t), \cdots, x_n(t)), \quad t \geqq 1, \tag{3}$$

$$\begin{aligned} y_i(1) &= y_{i\lambda}, \qquad \text{a specified starting value,} \\ z_j(t) &= g_j(y_1(t), y_2(t), \cdots, y_s(t); x_1(t), x_2(t), \cdots, x_n(t)), \quad t \geqq 1. \end{aligned} \tag{4}$$

The inputs, outputs, delays and combinational circuits are, in general, multi-valued; if they are all binary, the circuit is binary.

Every sequential circuit S defines a unique sequential machine M_S. If we define $X(t)$ to be the n-tuple, $X(t) = (x_1(t), x_2(t), \cdots, x_n(t))$, and similarly define $Y(t)$ and $Z(t)$, then we can find some σ and ρ such that:

$$Y(t+1) = \sigma(Y(t), X(t)), \tag{5}$$

$$Z(t) = \rho(Y(t), X(t)). \tag{6}$$

Thus the behavior of a sequential circuit is that of a sequential machine if we establish the correspondences: p with X, q with Y and r with Z. Note that a binary circuit is a special type of a machine with $P = \{0, 1\}^n$, $Q = \{0, 1\}^s$ and $R = \{0, 1\}^m$.

We conclude the introductory remarks by making precise the notion of the realization of a behavior (machine) by a given structure (circuit).

DEFINITION 3. A machine M *realizes* machine M' iff M' is a homomorphic image of a submachine of M (Hartmanis and Stearns, 1966).

DEFINITION 4. A circuit S realizes machine M iff the machine M_S, defined by S, realizes M.

III. FINITE MEMORY MACHINES

We assume that each combinational network and unit delay has an input side and an output side, and signals propagate from input to output.

DEFINITION 5. A sequential circuit is *feedback-free* if starting at any wire and passing through any number of connected devices, it is not possible to enter the same point twice.

DEFINITION 6. A reduced sequential machine is *definite* iff there exists an integer $k \geqq 0$ such that for all input sequences $\mathbf{x}$ with length $\geqq k$, $\sigma(q_i, \mathbf{x}) = \sigma(q_j, \mathbf{x})$ for all $q_i, q_j \in Q$.

It has been shown (Kleene, 1956; Simon, 1959; Arden, 1961; Brzozowski, 1962; McCluskey, 1962; Hartmanis and Stearns, 1963; Perles, Rabin and Shamir, 1963) that the machine defined by any feedback-free circuit is definite and that every definite machine can be realized by a feedback-free circuit. The behavior of any definite machine can be described by:

$$r(t) = \delta(p(t), p(t-1), \cdots, p(t-k)), \quad \text{for} \quad t \geqq k+1, \tag{7}$$

$$r(t) = \delta(p(t), p(t-1), \cdots, p(1), \quad \text{initial conditions}), \quad \text{for} \quad t < k+1. \tag{8}$$

If sequences of length $< k+1$ are of no interest, every definite machine is realizable by a single multi-valued shift register as shown in Fig. 2.

DEFINITION 7. A sequential machine is a *finite memory* (F. M.) machine if its output is only a function of the present input and a finite number of pasts inputs and outputs, i.e.,

$$r(t) = \psi(p(t), \cdots, p(t-k), r(t-1), \cdots, r(t-k)), \quad t \geqq k+1, \tag{9}$$

$$r(t) = \psi(p(t), \cdots, p(1), r(t-1), \cdots, r(1), \quad \text{initial conditions}), \quad t < k+1. \tag{10}$$

The describing equations suggest the realization (multi-valued) shown in Fig. 3, provided only the response to sequences of length $\geqq k+1$ is of interest. In this special case the circuit consists of two shift registers, one to remember the past inputs and one for the past outputs.

A more detailed account of F. M. machines can be found in Simon (1959), Gill (1962) and Brzozowski (1965). There is a decision p.ocedure

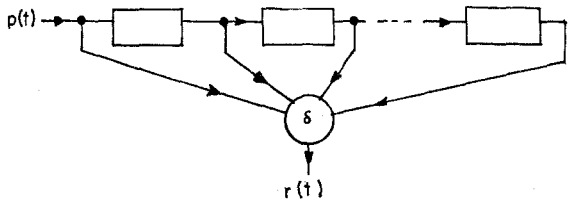


FIG. 2. Multi-valued feedback-free register circuit

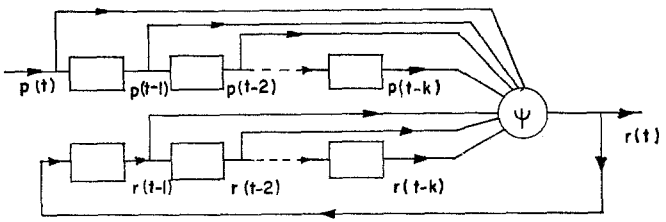


FIG. 3. Finite memory circuit

for testing whether a given machine has finite memory, and it is known that not all machines have F. M. However, we have the following general result:

LEMMA 1. *The state behavior of any sequential machine M can be realized by a multi-valued F. M. circuit.*

Proof. Consider the realization of any machine in the form of Fig. 1. Then if $q(t+1)$ is considered as an auxiliary output, $q(t+1) = r_a(t)$, we have

$$r_a(t) = \sigma(r_a(t-1), p(t)). \tag{11}$$

Thus the state behavior is F. M. from this point of view. The original output can be obtained by an additional combinational circuit,

$$r(t) = \rho(r_a(t-1), p(t)). \tag{12}$$

Of course the fact that the state behavior of any machine has F. M. character is of not much help to the circuit designer who attempts to get a simple structure for his realization. If one could use multi-valued logic then every machine could be realized using a single feedback loop carrying a v-valued signal. Since we are mostly limited to binary circuitry the designer must assign m binary variables y_j to represent q in such a way that $2^m \geqq v$. Making an arbitrary binary assignment will, in general,

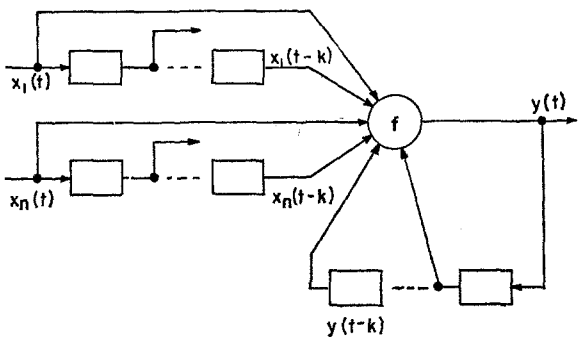


FIG. 4. F.M. circuit with one binary feedback loop

result in an undesired structure in which there are several feedback loops each carrying a binary signal.

Note that the F. M. circuit of Fig. 1, which realizes the state behavior of any machine, uses only a single delay for the delayed auxiliary output. Thus we are storing a *v-valued* signal for *one* unit of time. From the structural point of view one may prefer to store a *binary* signal for *several* units of time. This is possible if one can find an assignment for M which results in a circuit with a structure such as that shown in Fig. 4. For that circuit,

$$y(t) = f(y(t-1), \cdots, y(t-k); x_1(t), x_2(t), \cdots, x_n(t), \cdots, x_1(t-k), x_2(t-k), \cdots, x_n(t-k)), \tag{13}$$

and y is binary. If such a realization can be found for the state behavior then any output z_i is obtainable from the structure by means of one combinational circuit.

The general F. M. circuit which includes the response to short sequences can be represented as in Fig. 5, with the restriction that there is at least one unit delay in each feedback loop. Clearly this realization corresponds to Definition 7. Now consider a new variable $p'(t)$ whose alphabet is the product $P \times R$. Clearly in the corresponding representation shown in Fig. 5(b), C is *definite* for the augmented input alphabet. This suggests that the method for testing for definiteness is also applicable to testing for the F. M. property provided the augmented "input" is used. This is in fact true and this method can be found in Gill (1962) and McCluskey (1962).

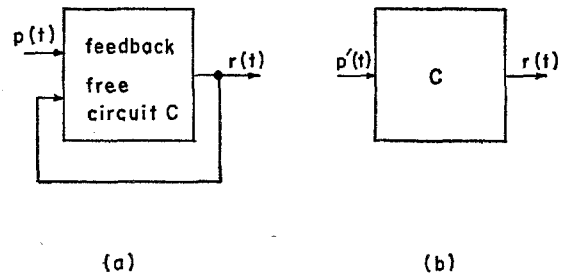


FIG. 5. General F.M. circuit

IV. FEEDBACK

Consider a given sequential circuit S, i.e., a given arrangement of combinational circuits and unit delays. Suppose we find a set W_i of k wires in S which, when cut, will remove all loops in S. Let the signal on the jth wire in the set be w_{ij} and let this signal be v_j-valued. In general there are many sets W_i of wires in a given circuit which, when cut, will remove all feedback. For each such set W_i define the *feedback value* F_i to be

$$F_i = \prod_{j=1}^{k} v_j . \tag{14}$$

The feedback value F of the circuit S can now be defined as the minimum F_i chosen from all possible F_i. More than one wire set may correspond to this minimum F.

THEOREM 1. *If a sequential circuit S has feedback value F, then there exists a total-state partition π (with F blocks) of the total states of S, which is such that an F-valued output r_a assigned according to π has F. M. property.*

Proof. [For background material on partitions refer to Hartmanis and Stearns (1966).] Since S has feedback value F, there exists a set of wires with signals $W = \{w_1, w_2, \cdots, w_k\}$ the cutting of which will remove all feedback loops, and $F = \prod_{i=1}^{k} v_i$. But every signal in S must be a function of only the present input and the present internal state. Thus

$$W(t) = \omega(X(t), Y(t)), \tag{15}$$

where $W(t)$ is considered either as a multi-valued signal or a k-tuple, and similar considerations apply to $X(t)$ and $Y(t)$. Now for a fixed total

state (X, Y), W is fixed and hence defines a total-state partition π on the total states of S. Clearly π has F blocks because W has F values, and two total states of S are in the same block if and only if they have the same W value.

Consider the internal state variable Y with all the wires in W cut. Since there is no feedback, the circuit is definite for the input X/W. Hence

$$Y(t+1) = \theta(X(t), X(t-1), \cdots, X(t-k), W(t), W(t-1), \cdots, W(t-k). \tag{16}$$

But

$$\begin{aligned} W(t) &= \omega(X(t), Y(t)) \\ &= \mu(X(t), X(t-1), \cdots, X(t-k-1), W(t-1), \cdots, W(t-k-1)). \end{aligned} \tag{17}$$

Hence W has F. M. property and any r_a assigned according to π has F. M. property.

This then establishes the basic relation between the feedback value of a circuit, and the F. M. property of its output.

V. SINGLE LOOP REALIZATIONS

As was said before, the problem of deciding whether a given sequential machine requires no feedback (is definite) has been solved. The next logical question is: "If a machine is not definite, how much feedback is required?" This has been answered recently by Friedman (1966) who showed that every indefinite sequential machine can be realized by a circuit with a single binary feedback loop. In view of our discussion of multi-valued realizations above, this means that instead of storing a multi-valued signal for one unit of time, it is possible to store a binary signal for several units of time.

The problem of finding single-loop realizations is the subject of this paper. Similar results were obtained independently by Friedman at about the same time.

We define a sequential circuit to be *single-loop* if it is not feedback-free, but there exists a wire (carrying a binary signal) which when cut renders the circuit feedback-free. The following results follow directly from Theorem 1.

COROLLARY 1. *A sequential circuit S is single-loop iff its state table is*

not definite and there exists a binary total state partition π on the states of S, such that any output assigned according to π has F. M. property. (For brevity, we shall simply say that π itself has the F. M. property.)

COROLLARY 2. *A sequential machine M is realizable by a single-loop circuit iff the flow table T of M or some expanded (state-split) version of T has a partition with F. M. property.*

We now consider the problem of testing whether a given flow table possesses a binary total-state feedback partition, i.e., one that has F. M. property.

VI. PARTITIONS WITH F.M. PROPERTY

Consider a flow table T with a multi-valued input $p \in P = \{p_1, p_2, \cdots, p_u\}$, internal states $q_1, q_2, \cdots, q_v$ and a binary output $r \in R = \{0, 1\}$.

DEFINITION 8. An input-output sequence, $\mathbf{s} = p(1)/r(1), p(2)/r(2), \cdots, p(t)/r(t)$, where $p(i) \in P$, $r(i) \in R$ is *applicable* to state q_i if and only if, when the input sequence $p(1), p(2), \cdots, p(t)$ is applied to T in state q_i, the resulting output sequence is $r(1), r(2), \cdots, r(t)$.

DEFINITION 9. The state q_j reached from q_i by an input-output sequence $\mathbf{s}$ applicable to q_i is called the $\mathbf{s}$-*successor* of q_i and is denoted by $q_j = \sigma_{\mathbf{s}}(q_i)$.

DEFINITION 10. An input-output sequence $\mathbf{s}$ *resolves* a set $Q_i = \{q_{i_1}, q_{i_2}, \cdots, q_{i_j}\}$ of states if and only if, for all $q_k \in Q_i$ for which $\mathbf{s}$ is applicable, $\sigma_{\mathbf{s}}(q_k)$ is the same.

As an example consider the flow table of Fig. 6. The sequence $\mathbf{s}_1 = (0/0, 0/1)$ is applicable to state C but not applicable to states A, B, D. Since the $\mathbf{s}_1$-successor of C is unique, $\mathbf{s}_1$ resolves the set $\{A, B, C, D\}$. On the other hand the sequence $\mathbf{s}_2 = (1/0, 1/0)$ is applicable to states A

	0	1
A	A 0	D 0
B	A 1	D 1
C	B 0	C 1
D	C 1	A 0

Machine M_1

FIG. 6. Machine M_1

and D, but does not resolve $\{A, D\}$ because the s_2-successor of A is A and that of D is D.

LEMMA 2. *The output of a sequential machine M has the F. M. property if and only if there exists an integer $k \geqq 0$ such that all input-output sequences of length k resolve the set Q of states of M. The lemma is a restatement of previous results* (Gill, 1962; Brzozowski, 1965).

LEMMA 3. *An input-output sequence* s *resolves the set Q of states if and only if* s *resolves all sets $Q_{ij} = \{q_i, q_j\}$, with $q_i, q_j \in Q$. The proof follows from the definitions.*

LEMMA 4. *The output r of a machine M has F. M. property if and only if there exists a $k \geqq 0$ such that all pairs $\{q_i, q_j\}$ of states of M are resolved by all input-output sequences of length k. This follows from the preceding lemmas.*

It is convenient to restate Definition 10 in a modified way:

DEFINITION 11. A pair $\{q_i, q_j\}$ of states is *resolved* if it is resolved by all sequences of length $\geqq k$, for some k.

In searching for a binary output with F. M. property we are going to examine all pairs of states and ensure that they are resolved. Now we assume that we know only the state behavior and that the output is to be found. For any pair of states $\{q_i, q_j\}$ and any input p we can only have two possibilities:

(1) $\sigma_p(q_i) = \sigma_p(q_j)$. In this case the pair $\{q_i, q_j\}$ is resolved by p/r and by all sequences beginning with p/r, where r can be either 0 or 1. Hence the total states (q_i, p) and (q_j, p) impose no restrictions on the output. In other words the output values assigned to these total states may be 0, 0 or 0, 1 or 1, 0 or 1, 1. In terms of the total state partition π for which we are searching, this pair of total states can appear in the same block of π or may be separated by π without affecting the resolving ability of an output assigned according to π.

(2) $\sigma_p(q_i) \neq \sigma_p(q_j)$. Here we distinguish two subcases:

(a) $\{q_i, q_j\} = \{\sigma_p(q_i), \sigma_p(q_j)\}$, i.e., the set $\{q_i, q_j\}$ and its p-successor set are identical. It is clear that π *must* separate the total states (q_i, p) and (q_j, p), for otherwise the pair $\{q_i, q_j\}$ will not be resolved by the arbitrarily long sequence $p/r, p/r, \cdots, p/r$, and hence r can't be F. M. In this case the examination of the two total states gives us a definite requirement on π: π must separate the total states. We will indicate this by $(q_i/_p\, q_j)$ which means that the transitions from q_i and q_j under p must be accompanied by *different* outputs, or that π separates total state (q_i, p) from (q_j, p).

(b) $\{q_i, q_j\} \neq \{\sigma_p(q_i), \sigma_p(q_j)\} = \{q_k, q_l\}$. Here we have two possibilities. Either $(q_i/_p q_j)$, or $\{q_k, q_l\}$ must be resolved. We will discuss this case more fully after a graphical aid is introduced.

We shall first construct the *pair-table* $P(T)$ of a flow table T. The pair-table will have the appearance of a flow table except that the present states correspond to pairs of states of T and the next states are the successor sets of these pairs. This is best illustrated by an example. The pair-table of M_1 in Fig. 6 is shown in Fig. 7. The next-state entries are either pairs or single states. Since we are looking for restrictions on the partition π we can remove those pairs which do not demand any restrictions. For example, if the next-state entry in the pair table is a single state, we can remove this entry, for the output for the corresponding pair of total states is optional. This is illustrated in Fig. 8(a), where the next state entries for AB are removed. Now it is clear that AB is resolved. Since AC depends on AB under input 0 and AB is resolved, then AC will also be resolved by all sequences beginning with 0. Hence the next state entry AB for AC under 0 is removed, and also the AB entry is removed from row BC. There are no further simplifications of this type which we will call *sink simplifications*. This gives Fig. 8(b).

Note however in Fig. 8(b) that the pair BD does not appear as a next-state entry. Hence the resolution of the pairs does not depend on the resolution of BD. Hence we can also remove BD provided AC and AD are resolved, for the resolution of these successor pairs of BD guarantees the resolution of BD. This second type of simplification will be called *source simplification*. If no further simplifications of either type are possible the pair table will be called a *stripped* pair table shown in Fig. 8(c).

The above process has a nice interpretation in terms of signal flow graphs (Mason, 1953; Brzozowski and McCluskey, 1963). Let us represent the pair table of Fig. 7 as a pair graph, without showing any transitions leading to single states. We obtain the diagram of Fig. 9. We note that the node AB is a sink node (has no outgoing transitions). Since this corresponds to AB being resolved, the node and all transitions into it can be removed. Hence we have chosen sink simplification as a suitable name. Further, note that node BD is a source (no incoming transitions); hence it can be removed together with its outoing transitions. The stripped pair diagram is shown in Fig. 10. It is clear that the stripped pair diagram has the property that every node appears in at least one closed loop. In other words, all cascade nodes have been removed and only feedback nodes remain.

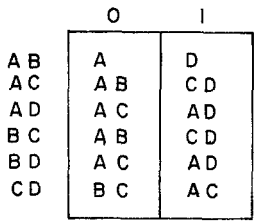


FIG. 7. Pair table for M_1

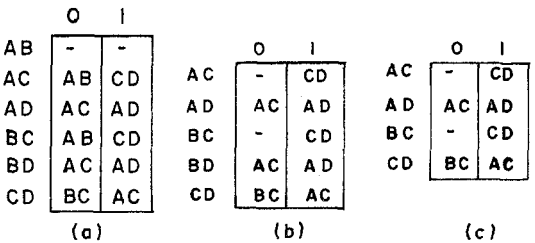


FIG. 8. Steps in simplifying the pair table

We can now use the stripped pair graph to find all the restrictions on π, the total state partition for which we are searching. It is clear that the output assigned according to π must remove all loops in the pair diagram, otherwise some pair of states will not be resolved. (This result has also been found by Friedman.) In the specific example of Fig. 10, it is clear that the following separations must be achieved: $A/_1 D$ and (either $A/_1 C$ or $C/_1 D$) and (either $B/_1 C$ or $C/_0 D$). We shall replace the "and" connective in the above statement by a dot which will be omitted and the "either or" by a $\vee$. The corresponding algebraic statement will be called the generating function of π and denoted by $\gamma(\pi)$. Here

$$\gamma(\pi) = (A/_1 D)(A/_1 C \vee C/_1 D)(B/_1 C \vee C/_0 D).$$

We can now perform algebraic manipulations on $\gamma(\pi)$ in order to put it in a more useful form. First the expression can be multiplied out to give:

$$\gamma(\pi) = (A/_1 D)(A/_1 C)(B/_1 C) \vee (A/_1 D)(A/_1 C)(C/_0 D) \vee (A/_1 D)(C/_1 D)(B/_1 C) \vee (A/_1 D)(C/_1 D)(C/_0 D);$$

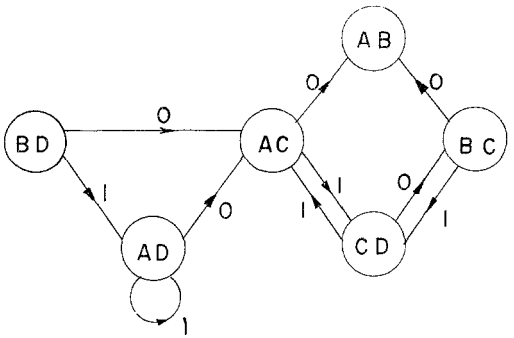


FIG. 9. Pair graph for M_1

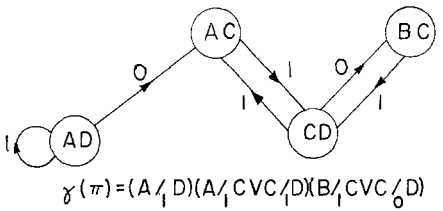


FIG. 10. Stripped pair graph for M_1

clearly this is a valid restatement of the above conditions. Now each product in the sum represents a specific set of sufficient conditions on π. For any such product we examine the 0 separations separately from the 1 separations for we are dealing with two different columns of the flow table. For example the first term states that π must separate total states $(A, 1)$ from $(D, 1)$, $(A, 1)$ from $(C, 1)$ and $(B, 1)$ from $(C, 1)$. If this is done, no restrictions in the 0-column are needed. The set of conditions in a given product which correspond to one input can be put in a more convenient form by further multiplication. Let Q_1, Q_2, Q_3, Q_4 be sets of internal states of sequential machine. Further suppose that at some stage we reach the conclusion that π must separate Q_1 from Q_2 and Q_3 from Q_4, both separations to be done under the same input. We assume $Q_1 \cap Q_2 = Q_3 \cap Q_4 = \varnothing$, where $\varnothing$ is the empty set. In other words we have

$$(Q_1/Q_2)(Q_3/Q_4),$$

where the input p is dropped for convenience. It is clear that this can be

done either by the separation:

$$(Q_1 \cup Q_3)/(Q_2 \cup Q_4),$$

provided

$$(Q_1 \cup Q_3) \cap (Q_2 \cup Q_4) = \varnothing,$$

or by

$$(Q_1 \cup Q_4)/(Q_2 \cup Q_3),$$

provided

$$(Q_1 \cup Q_4) \cap (Q_2 \cup Q_4) = \varnothing.$$

If neither one of these conditions is satisfied, then there does not exist a partition corresponding to the separation requirements.

We illustrate these manipulations with the first term of $\gamma(\pi)$:

$$((A/D)(A/C))(B/C) = (A/CD)(B/C) = AB/CD.$$

Continuing this simplification for other terms, we obtain:

$$\gamma(\pi) = (AB/_1\, CD) \vee (A/_1\, CD)(C/_0\, D) \vee (AC/_1\, BD) \vee (AC/_1\, D)(C/_0\, D).$$

If a separation requirement in the modified $\gamma(\pi)$ involves all the states of the machine then we have one partition corresponding to the requirement. For example $(AB/_1\, CD)$ represents the partition $\{\overline{A, B}; \overline{C, D}\}_1$, where the subscript 1 indicates that the partition is on total states involving input 1. Since no requirements on the 0 column are needed, *any* partition in this column will do. On the other hand if some states are not involved in the separation, they may be put in either block. Thus $(A/_1\, CD)$ generates either $\{\overline{A, B}; \overline{C, D}\}_1$ or $\{\overline{A}; \overline{B, C, D}\}_1$.

In summary, the generating function describes *all* total state partitions (which are now treated as pairs, in the case of two inputs, of internal state partitions, for convenience), which can be used for feedback. For example, we may choose $\pi_1 = \{\overline{A, D}; \overline{B, C}\}_0, \{\overline{A, B}; \overline{C, D}\}_1$, which satisfies the requirements $(C/_0\, D)(A/_1\, CD)$. The flow table of M_1 with output assigned according to π_1 is shown in Fig. 11. It is easy to verify that the output has F.M. property. Our original output for M_1 in Fig. 6 used the partition: $\{\overline{A, C}; \overline{B, D}\}_0, \{\overline{A, D}; \overline{B, C}\}_1$ which does not satisfy any of the separation requirements of $\gamma(\pi)$; hence it cannot be used for feedback.

	0	1
A	A,0	D,1
B	A,1	D,1
C	B,1	C,0
D	C,0	A,0

FIG. 11. An F.M. output for M_1

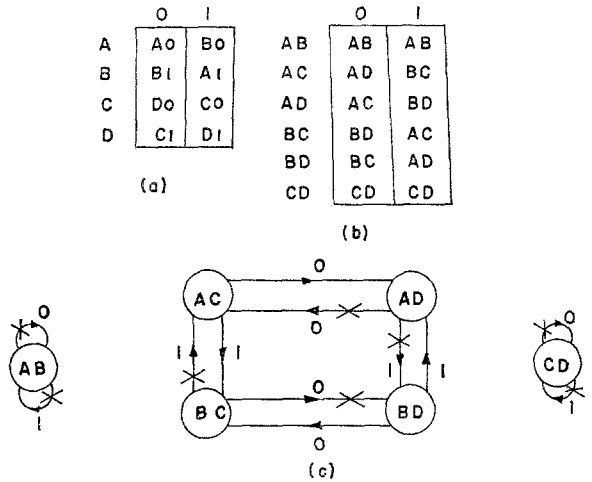


FIG. 12. (a) M_2. (b) Stripped pair table for M_2. (c) Stripped pair graph for M_2.

Another example is shown in Fig. 12. The generating function is:

$$\begin{aligned}\gamma(\pi) = {} & (A/_0 B)(A/_1 B)(C/_0 D)(C/_1 D) \\ & [(A/_0 C) \vee (A/_0 D)][(B/_0 C) \vee (B/_0 D)] \\ & [(A/_1 C) \vee (B/_1 C)][(A/_1 D) \vee (B/_1 D)] \\ & [(A/_0 C) \vee (A/_1 D) \vee (B/_0 D) \vee (B/_1 C)] \\ & [(A/_0 D) \vee (A/_1 C) \vee (B/_0 C) \vee (B/_1 D)].\end{aligned}$$

This is a rather lengthy expression and could be evaluated by a computer program. For hand computation partial multiplication will very often reveal the answer. From the first and third expression we have $[(AC/_0 BD) \vee (AD/_0 BC)]$, and from second and fourth terms $[(AC/_1 BD) \vee (AD/_1 BC)]$. The pair graph is a very convenient aid.

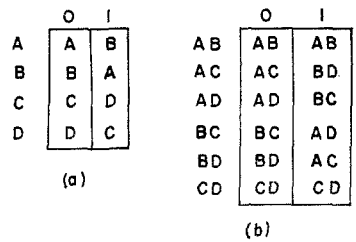


FIG. 13. (a) M_3 . (b) Pair Table for M_3

If we choose $(AC/_0\, BD)$ then all loops involving 0 input only are broken, as shown by crosses on the corresponding transitions in Fig. 12(c). There is still a loop $(AC) \rightarrow^0 (AD) \rightarrow^1 (BD) \rightarrow^0 BC \rightarrow^1 AC$. Hence in the 1 column we must separate $(A/_1\, D)$ or $(B/_1\, C)$. Choose $(AC/_1\, BD)$; Thus the partition $\pi = \{\overline{A, C}; \overline{BD}\}_0 \{\overline{AC}; \overline{BD}\}_1$ will result in an F. M. output.

Finally, we give an example for which π does not exist. Consider M_3 of Fig. 13(a). The pair graph of Fig. 13(b) reveals that every pair of states must be separated under input 0. Clearly this is impossible. Hence state splitting must be applied.

SUMMARY OF GENERAL PROCEDURE

Given a flow table of a sequential machine M:

1. Remove all the given outputs of M from the flow table.
2. Draw the pair graph of M.
3. Obtain the stripped pair graph of M by removing all the cascade nodes (nodes not appearing in any closed loop) and all the branches originating or terminating at cascade nodes.
4. *If the stripped pair graph is empty, then M is definite.* List all the simple loops in the stripped pair graph. By a simple loop we mean a loop in which no node appears more than once.
5. For each simple loop list all the separations each of which will break the loop. This gives a term of the form $(q_1/_{p_1}\, q_2 \vee q_3/_{p_2}\, q_4 \vee \cdots)$, which is a sum of separations, for breaking any link will break the loop.
6. The generating function $\gamma(\pi)$ is the product of all the terms obtained in Step 5.
7. Multiply out $\gamma(\pi)$ to obtain a sum of products form for $\gamma(\pi)$. (Each separation still involves only two states.)
8. Check each product of Step 7 under each input (separately) to

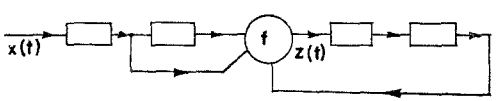
determine whether the separation requirements can be satisfied. This can be done by inspection or formally as in Step 9. *If the separation requirements cannot be satisfied for any product in* $\gamma(\pi)$ *then there is no partition with F.M. property for M.*

9. "Multiply out" each product to obtain separations involving more than two states until each product in the sum of products in $\gamma(\pi)$ involves only one separation per input. If in this formal multiplication one or more states appear on both sides of the separating line the entire product is rejected.

10. Having obtained $\gamma(\pi)$ in the form of Step 9, if $\gamma(\pi)$ is not empty, then *any partition satisfying one or more of the terms in* $\gamma(\pi)$ *has F.M. property for M, and any partition with F.M. property must satisfy at least one term of* $\gamma(\pi)$.

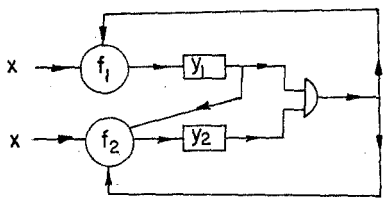
VII. REALIZATIONS FROM F.M. PARTITIONS

Once a partition π with F.M. property has been found, a corresponding one-loop realization follows. First assign an output z according to π, then realize the machine by using only a finite number of past values of z and of the inputs. See Brzozowski (1965) or Hartmanis and Stearns (1966) for further details. It is clear that a variety of realizations result from a given generating function, in general. For example, for M_1 of Fig. 6, (the example was taken from Hartmanis and Stearns (1963) using $\{\overline{A, B, C}; \overline{D}\}_0$, $\{\overline{A, B, C}; \overline{D}\}_1$ and Gill's (1962) method for finding the assignment we obtain a circuit of the form of Fig. 14. It can be easily verified that no delays can be removed, if one wants to maintain the circuit in the finite memory form using z as the F.M. output. Thus 4 delays are required since $z(t)$ depends on $x(t-2)$ and $z(t-2)$. Yet Hartmanis and Stearns (1963) have found a realization for M_1 which has one-loop, only 2 delays, and the same feedback partition, as shown in Fig. 15. Furthermore, even if we restrict the realizations to only those in F.M. form, different amounts of input and output memories may be required depending on the partition used. Notice that in order to find



$$z(t) = x(t-1)[\,\bar{x}(t-2)\,\bar{z}(t-2) + x(t-2)\,z(t-2)]$$

FIG. 14. An F.M. realization of M_1



$$y_1' = \bar{x}\, y_1\, y_2 + x\, \overline{y_1\, y_2} \qquad y_2' = \bar{y}_1 + x\, y_1\, y_2$$

FIG. 15. Another realization of M_1

the realization of Fig. 14, we have only used partition methods in the sense that the flow table was not expanded to determine the feedback partitions. Yet the F.M. form in fact does result in state splitting (several states of the delays in Fig. 14 represent each state of M_1). However, this state splitting arises in a natural way, since the split states are easily found.

It is up to the designer to determine which circuit form is better; at any rate he has many single-loop circuits available to choose from.

VIII. THE GENERAL PROBLEM

If a given flow table T does not have an F.M. partition it is necessary to examine expanded versions of T. Friedman's results ensure that there always exists an expanded flow table having an F.M. partition. However, his method is not economical and the problem of finding economical single-loop circuits requires further work.

Perhaps the most immediate step is to attempt to expand the table by state-splitting using set-systems (Hartmanis and Stearns, 1966) rather than partitions on the given table. This does not appear to be an easy method, as will be illustrated by an example. Figure 16 shows an autonomous automaton M_4 with 4 states, for which no partition exists, because every pair of states must be split. The table is completely symmetric so we can start with any state. First, A can be separated from all other states by the partition $\{\overline{A}; \overline{B, C, D}\}$. Next, in the second block, B must be separated from C and D; hence try the set system $\{\overline{A, C, D}; \overline{B, C, D}\}$. Now the split states will be given different assignments. In order to complete the expanded table, label the split states by different subscripts. Thus $\{\overline{A, C_0, D_0}; \overline{B, C_1, D_1}\}$ is the trial partition in the

A	A
B	B
C	C
D	D

FIG. 16. M_4

A	A	0		A	A	0
B	B	1		B	B	1
C_0	C	0		C_0	C_1	0
C_1	C	1		C_1	C_0	1
D_0	D	0		D_0	D_1	0
D_1	D	1		D_1	D_0	1
	(a)				(b)	

FIG. 17. (a) Expansion of M_4 . (b) Assignment of subscripts

expanded table. Since we are dealing with an autonomous case, internal-state partitions and total-state partitions coincide; let $z = 0$ for the first block $-z = 1$, for the second. We obtain the expanded table of Fig. 17(a). Notice that to each set system there correspond several expanded tables, for we have yet to assign subscripts in the next state column of Fig. 17(a), and this can be done in several ways. If the partition can be used for feedback, the next state entry for C_0 cannot be C_0 , for the pair AC_0 would not be resolved. Hence it must be C_1 . Using such arguments we obtain the table of Fig. 17(b). We are still left with a cycle (C_0 , D_0), (C_1 , D_1). In this case there is no successful assignment of subscripts; hence the set system is unsuitable. Should we now try another set system for the original table, or try to split more states in the expanded table? The procedure here is not clear. The simplest expansion that we have been able to find for M_4 requires that two of the states be split 3 ways as shown in Fig. 18. Since the set system approach appears somewhat uncertain at this stage other means are used to shed more light on the problem.

IX. CONSTANT INPUT MACHINES

In this section we consider only machines with a constant input (autonomous case), i.e., flow tables with a single next state column.

DEFINITION 12. A state of a constant input machine is called *transient* if it does not appear in a cycle. It is called *cyclic* otherwise. A sequential machine is cyclic if it contains only cyclic states.

A	A	0
B	B	1
C_1	C_2	0
C_2	C_3	0
C_3	C_1	1
D_1	D_2	0
D_2	D_3	1
D_3	D_1	1

FIG. 18. A successful expansion of M_4

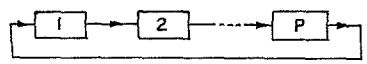


FIG. 19. Universal constant-input cyclic circuit

THEOREM 2. *Every cyclic constant input machine can be realized by a circuit of the form shown in Fig. 19.*

Proof. Let the flow table contain cycles of lengths $l_1 < l_2 < \cdots < l_n$. Find the least common multiple L of $(l_1, l_2, \cdots, l_n)$; then each l_i divides L. Now choose an integer p as follows: If the total number K of cycles is less than $L - 1$, let $p = L$. If $K \geqq L - 1$, choose the smallest N such that $LN > K + 1$ and let $p = LN$. Now we have a p such that each l_i divides p and $K < p - 1$. Now consider the circuit U, with p chosen as above. If the circuit is started in any state which has m adjacent 1's, $0 < m < p$, and 0's elsewhere, then it will cycle through all p states with m adjacent 1's. Take the first cycle of length l_1 in the given flow table. Expand this cycle to a cycle of length p; this is possible since l_1 divides p. Now all the states which are separated by distance l_1 in the expanded cycle are equivalent to the same state in the original cycle. Assign any state of U with $m = 1$ to represent the first state of the given cycle. It is clear that the first cycle can be realized. Similarly assign $m = 2$ to the second cycle $-m = 3$ to the third cycle, and finally $m = K$ to the last cycle. We have chosen $p > K + 1$; hence all cycles are realized. This concludes the proof.

It is interesting to note that U does not require any combinational logic whatsoever.

THEOREM 3. *Every constant input machine can be realized by a circuit consisting of a universal circuit and of a separate feedback-free circuit.*

Proof. We shall use a construction suggested to us by W. A. Davis.[2]

[2] **Personal communication.**

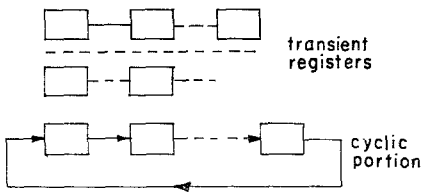


FIG. 20. Logic-free, one-loop realization of a constant-input machine

Basically, the cyclic portion will be realized by the universal circuit U, and transient states by added delays. Suppose q_1 is a state which does not appear in the next-state column. Then if the machine is started in q_1 it will pass through transient states $q_1, q_2, \cdots, q_i$, and then must enter into a cycle, beginning with state q_{i+1}. For each such set of states $q_1, q_2, \cdots, q_i$ assign a shift register of i delays. Now, if the automaton is to be started in a cyclic state, set all transient delays to 0. The behavior will be as in Theorem 2. If it is desired to start in a transient state, q_k, put a 1 in any transient delay representing this state. (The first state q_1 will have a unique delay, but other states may appear in more than one transient register). Furthermore, if the cycle is entered from q_k after j steps, (i.e., state q_{i+1} is reached) start the cyclic portion in the jth predecessor of q_{i+1} in the cycle. Thus, by the time the single transient 1 disappears out of the transient register, the cyclic part is in the proper cyclic state.

This concludes the construction, the form of which is shown in Fig. 20. Note that again no *combinational logic* is required. Furthermore the circuit is linear in a trivial way.

One can save a few delays at the expense of a few exclusive or gates. This can be done in the transient portion as follows: Assign one delay for each state which has no predecessor. Call this set of states and delays rank 1. If a successor q_1 of a rank 1 state has two (or more) predecessors q_2, q_3, the input to the delay of q_1 is the modulo two sum of q_2 and q_3 delay outputs. It is easy to verify that this construction will give the proper operation.

CONCLUSIONS

For machines with inputs it is also possible to exhibit a universal one-loop structure. This approach due to Hopcroft and (independently) McNaughton is described by Friedman (1966) and the reader is referred to that paper for details.

We have presented methods of finding single-loop realizations of machines. It is felt that these realizations will be very useful to the general assignment problem since they represent a class of special structural forms and should be considered when the choice of assignment is being made.

RECEIVED: September 9, 1966

REFERENCES

ARDEN, D. N. (1961), Delayed logic and finite state machines. *Proc. 2nd Ann. Symp. Switching Circuit Theory Logical Design*, Sept. 1961, pp. 1–35. Chicago, Ill.

BRZOZOWSKI, J. A. (1962), Canonical regular expressions and minimal state graphs for definite events. *Proc. Symp. Math. Theory Automata, 12th*, pp. 529–561. Polytechnic Press, Microwave Research Institute Symp. Series., Polytechnic Inst. of Brooklyn, N.Y.

BRZOZOWSKI, J. A. (1965), An essay on feedback. Dept. of Elec. Eng., The University of Ottawa, Ottawa, Canada, Tech. Rept. No. 65-2; March, 1965.

BRZOZOWSKI, J. A., AND MCCLUSKEY, E. J., (1963), Signal flow graph techniques for sequential circuit state diagrams. *IEEE Trans. Electron. Computers*, **EC-12,** 67–76.

GILL, A. (1962), "Introduction to the Theory of Finite-State Machines." McGraw-Hill, New York.

HARTMANIS, J., AND STEARNS, R. E. (1963), A study of feedback and errors in sequential machines. *IRE Trans. Electron. Computers* **EC-12,** 223–232.

HARTMANIS, J., AND STEARNS, R. E. (1966), "Algebraic Structure Theory of Sequential Machines." Prentice Hall, Englewood Cliffs, New Jersey.

FRIEDMAN, A. D. (1966), Feedback in synchronous sequential switching circuits. *IEEE Trans. Electron. Computers* **EC-15,** 354–367.

KLEENE, S. C. (1956), Representation of events in nerve nets and finite automata. *In* "Annals of Mathematical Studies," Vol. 34, "Automata Studies" pp. 3–41. Princeton University Press, Princeton, N.J.

MASON, S. J. (1953), Feedback theory—some properties of signal flow graphs. *Proc. IRE*, **41,** 1144–1156.

MCCLUSKEY, E. J. (1962), Reduction of feedback loops in sequential circuits and carry leads in iterative circuits. *Proc. 3rd Ann. Symp. Switching Circuit Theory Logical Design*, pp. 91–102. Chicago, Ill., Oct. 1962.

PERLES, M., RABIN, M. O., AND SHAMIR, E. (1963), The theory of definite automata. *IEEE Trans. Electron. Computers* **EC-12,** 233–243.

SIMON, J. M. (1959), A note on memory aspects of sequence transducers. *IRE Trans. Circuit Theory* **CT-6,** 26–29.