Uniform Synthesis of Sequential Circuits†

By J. D. ULLMAN and PETER WEINER‡

(Manuscript received July 19, 1968)

In this paper we consider the synthesis of sequential machines by networks of a fixed module with delay. We show that every binary input n *state sequential machine has an isomorphic realization using at most* p *copies of a module with* $2r + 1$ *inputs, where* p *is the smaller of* $\frac{2r}{2r-1}(n^{1+\log_r 2} + 4n^{1+\log_r 4})$ *and* $r2^{\lceil n/r \rceil}$. ($[x]$ *is the smallest integer* $\geqq x$.)

I. INTRODUCTION

The realization of an arbitrary binary output synchronous sequential machine by a network of copies of a fixed sequential machine (module) or copies of a small number of machines is a problem which has received recent attention.[1-5] An equivalent problem has been studied in Ref. 6. A design of this sort is particularly suited to batch fabrication techniques, because it is possible to mass produce a fairly complex integrated circuit (the module) and then wire these circuits together to realize any desired sequential machine.

The machines, so constructed, will be fast; the time between inputs need not be longer than the time it takes a single module to resolve its output after a change in input, no matter how many modules are in the network. The disadvantage of this technique, so far, has been the large number of copies of the module necessary to realize a machine; as many as $2^n - 2$ copies for an n state machine are required when using the modules of Refs. 1 and 2. These modules are shown in Fig. 1 for the binary input case.

Not shown in any of our diagrams is provision for initializing the output of any module to the hot (1) state if desired. Neither is provision for control of the module by a clock shown in this or any other module.

† Portions of this paper appeared in the Proceedings of the IEEE 9th Annual Symposium on Switching and Automata Theory, Schenectady, N. Y., October 1968.

‡ Princeton University, Princeton, New Jersey.

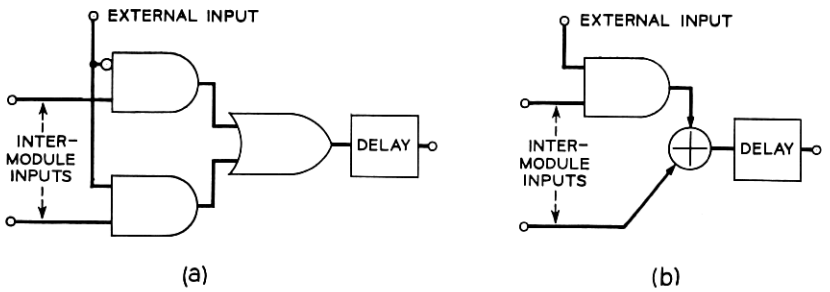

Fig. 1 — Simple modules.

The modules of Fig. 1 each have two intermodule inputs, that is, inputs to which either logical constants or the output of some module will be connected. If a module with two intermodule inputs is universal (can realize any sequential machine having one binary input), then there is a unique minimal network composed of copies of this module realizing a particular sequential machine with a binary input.[1,2] If there are more than two intermodule inputs, there may be more than one network realizing a given machine. We consider a class of modules with different numbers of intermodule inputs and attempt to design small networks consisting of copies of one of the modules in the class.

The class of modules we use for single input machines is represented schematically in Fig. 2a. There is a member of the class with $2r$ intermodule leads for each $r \geqq 1$. Let the module of Fig. 2a with a particular value of r be M_r. Note that M_1 is essentially the same as the module of Fig. 1a. M_2 is shown in Fig. 2b.

In what follows, we restrict ourselves to the design of networks for the realization of machines with one binary input. The generalization to the use of machines having k binary inputs is straightforward when one uses a class of modules represented schematically in Fig. 3.

Notice that conventional designs of sequential circuits, represented schematically in Fig. 4, require the construction of $\log_2 n$ Boolean functions of $k + \log_2 n$ variables, where k and n are the number of input variables and states, respectively, of the machine. The number of gates necessary for a two-level realization of several functions of p variables can be as high as 2^p, so one would expect, even in the case $k = 1$, to require as many as n gates for a realization in the form of Fig. 4. We cannot show, for fixed r, that all n state machines with

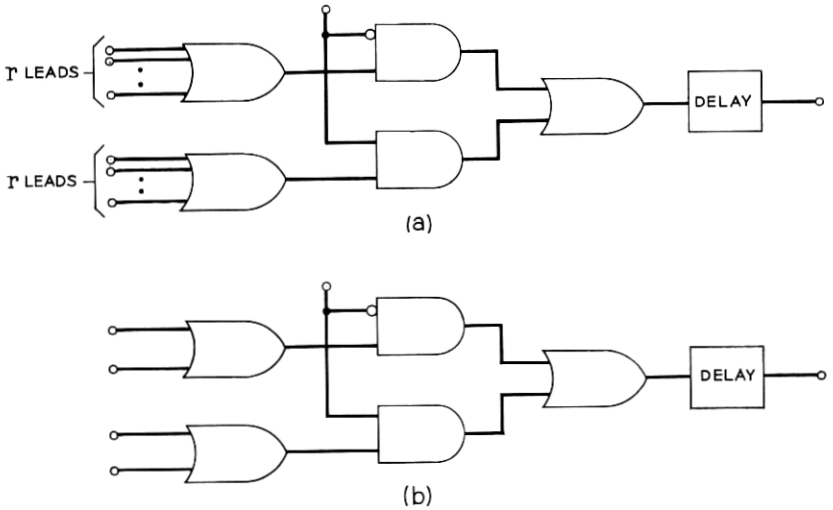

Fig. 2 — (a) The module M_r; (b) the module M_2.

single inputs can be realized by networks of as few as n copies of M_r. However, we show that the number of copies of M_r needed to realize any binary input n state sequential machine is bounded above by two functions of n. These functions, to within a constant factor, are $2^{n/r}$ and $n^{1+\log_r 4}$.

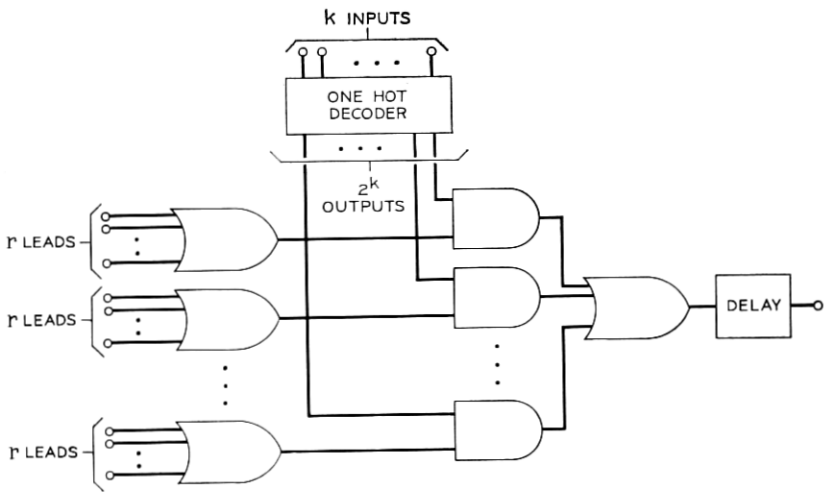

Fig. 3 — Generalization of M_r.

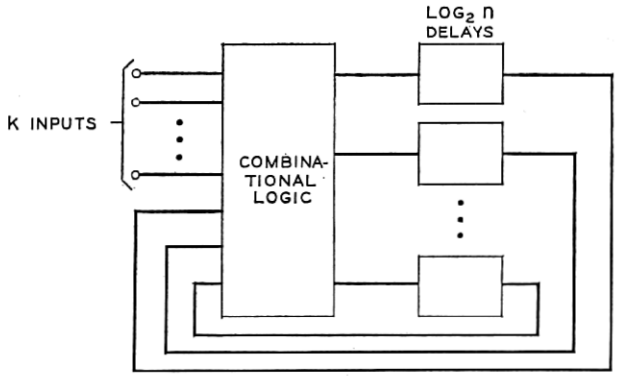

Fig. 4 — Conventional sequential circuit.

II. DEFINITIONS AND BASIC CONCEPTS

A sequential machine will be denoted $A = (K, \Sigma, \delta, q_0, F)$. K and Σ are finite sets of *states* and *inputs*, respectively. F, the *final states*, is a subset of K. It is the set of states for which the output is 1. q_0, the *start state* is a particular element of K. δ maps $K \times \Sigma$ to K. It gives the next state for each combination of state and input symbol. The function δ is usually displayed as a flow table, with a row for each state and a column for each input. The entry in the ith row and jth column is the value of δ for the ith state and jth input. The first state will always be the start state. An example is shown in Table I.

We extend δ to domain $K \times \Sigma^*$ by:†

(*i*) $\delta(q, \epsilon) = q$ for all q in K.
(*ii*) $\delta(q, wa) = \delta\ (\delta\ (q, w), a)$, for all q in K, w in Σ^*, and a in Σ.

The *event defined by* the machine A, denoted $T(A)$, is $\{w \mid \delta\ (q_0, w)$ is in $F\}$. That is, $T(A)$ consists of exactly those input strings which cause A to go from the start state to a final state. For example, if 3 and 5 are the final states of the machine of Table I, then 110 is in $T(A)$, since $\delta(1, 1) = 6$, $\delta(6, 1) = 4$ and $\delta(4, 0) = 5$. 001 is not in $T(A)$ since $\delta(1, 0) = 3$, $\delta(3, 0) = 1$ and $\delta(1, 1) = 6$.

Let R be a subset of Σ^* for some finite set Σ. For each w in Σ^*, define the *derivative* of R with respect to w, denoted R/w to be set of strings x such that xw is in R.‡

† Σ^* is the set of all strings of symbols in Σ, including ϵ, the string of length 0.
‡ This notion of derivative is "backwards" from that used in Ref. 7. It is actually the quotient operation of Ref. 8.

Let $A = (K, \Sigma, \delta, q_0, F)$ be a machine, and let $\Sigma = \{0,1\}$. We can define two "inverses" of δ, denoted μ_0 and μ_1. These functions map sets of states to sets of states by:

$$\mu_0(G) = \{q \mid \delta(q, 0) \text{ is in } G\},$$

$$\mu_1(G) = \{q \mid \delta(q, 1) \text{ is in } G\}.$$

For each subset G of K, let A_G be the machine $(K, \{0, 1\}, \delta, q_0, G)$. Let $H = \mu_0(G)$ and $J = \mu_1(G)$. If $R = T(A_G)$, then $R/0 = T(A_H)$ and $R/1 = T(A_J)$. For w is in $R/0$ if and only if $w0$ is in $T(A_G)$. But $w0$ is in $T(A_G)$ if and only if $\delta(q_0, w)$ is a state p such that $\delta(p, 0)$ is in G. Equivalently, w is in $R/0$ if and only if $\delta(q_0, w)$ is in H. The argument for $R/1$ is analogous.

When talking about a fixed sequential machine, $A = (K, \{0, 1\}, \delta, q_0, F)$, we often identify $T(A_G)$ with G for each subset G of K. We use $G/0$ and $G/1$ for $\mu_0(G)$ and $\mu_1(G)$. For example, if A is the machine of Table I and $G = \{1, 3, 5\}$, then $G/0 = \{1, 2, 3, 4, 5\}$ and $G/1 = \{3\}$.

A *network* of a module M is an interconnection of copies of M such that each intermodule input is connected to either the output of a copy of M in the network or a logical constant (0 or 1). The external inputs of each copy of M (or corresponding external inputs if a copy has more than one) are connected together and receive the input to the network. One copy of M is designated the output of the network; the network accepts an input sequence if the output of the designated copy is hot (1) after receiving the sequence.

The module M_2 of Fig. 2b is repeated as Fig. 5 with certain points marked. Suppose that this module is part of a network realizing the event F of the sequential machine $A = (K, \{0, 1\}, \delta, q_0, F)$. Suppose also, that it has been determined that the output of this copy of the module must be some event $G \subseteq K$. That is, the output of this module

TABLE I—NEXT STATE FUNCTION

		inputs 0	1
states	1	3	6
	2	5	4
	3	1	5
	4	5	6
	5	5	2
	6	2	4

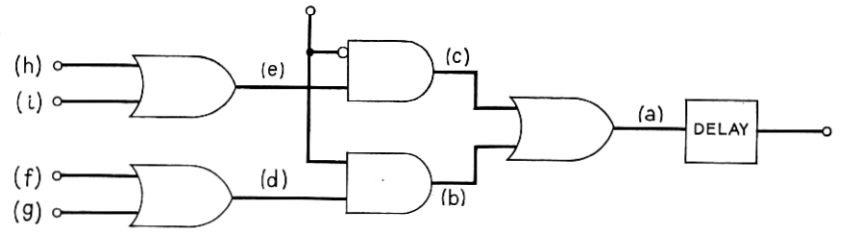

Fig. 5 — Points in the module M_2.

is hot exactly when the sequence of inputs to the network is in the event G. Thus, when the last input appears at the external input of this module, point a, the input to the delay, must immediately become hot if and only if the last input completes a sequence in G.

Observe that point b can be hot only if the last input is 1 and point c can be hot only if the last input is 0. Thus, immediately before the last input appears at the external input terminal, point d must be hot if and only if the previous inputs form a sequence in $G/1$ and point e must be hot if anly only if the previous input sequence is in $G/0$.

The union of the events at f and g must thus be $G/1$ and the union of events at h and i must be $G/0$. We are free to choose the events at the intermodule inputs subject only to these constraints. For example, we could choose the events at f and g to be those strings in $G/1$ of even and odd length, respectively. However, we restrict our choice so that the events at the intermodule leads will be representable as sets of states of A.

Design, using the module M_r, $r > 2$, proceeds the same way. If a given copy of the module is to realize the event G, then the lowest r of the intermodule inputs must be from modules realizing events $H_1, H_2, \cdots, H_r$ whose union is $G/1$; the remaining r intermodule inputs must be from modules realizing events $J_1, J_2, \cdots, J_r$, whose union is $G/0$. However, some of $H_1, \cdots, H_r$ or $J_1, \cdots, J_r$ may be the empty set or the set of all states, in which case these events are "realized" by logical constants rather than modules.

The above arguments justify the following reduction in the design problem for the class of modules M_r, $r \geqq 1$:

Let $A = (K, \{0, 1\}, \delta, q_0, F)$ be a sequential machine. An *M_r-synthesis* of A is a set $\mathcal{S}$ of subsets of K having the properties:

(*i*) F is in $\mathcal{S}$.

(*ii*) If G is in $\mathcal{S}$, then there are sets $H_1, H_2, \cdots, H_r$ and

$J_1, J_2, \cdots, J_r$ in $\mathcal{S}$, not necessarily all distinct, such that

$$\bigcup_{i=1}^{r} H_i = G/1 \quad \text{and} \quad \bigcup_{i=1}^{r} J_r = G/0.$$

From what we have said concerning the flow of signals in the module M_r, we may conclude that if $\mathcal{S}$ is an M_r-synthesis of A, then there is a network of m copies of M_r realizing $T(A)$, where m is the number of elements of $\mathcal{S}$ that are neither Φ nor K.† We call m the *size* of $\mathcal{S}$.

Notice that an M_r-synthesis requires that all modules realize events which are identifiable with a set of states. Such networks are called *isomorphic* to A. There may be networks of copies of M_r which realize $T(A)$, but are not M_r-syntheses of A. However, in our search for small networks we shall not consider any networks except those which are M_r-syntheses. See Ref. 5 for some comments on the existence of non-isomorphic realizations of sequential machines.

III. CONSTRUCTION OF M_r-SYNTHESES

The purpose of this paper is to show that M_r-syntheses of small size exist for an arbitrary n-state sequential machine. The first bound on the size of an M_r-synthesis is straightforward.

Let $A = (K, \{0, 1\}, \delta, q_0, F)$ be an n state sequential machine. We may choose r disjoint subsets of K, say $K_1, K_2, \cdots, K_r$, such that $\bigcup_{i=1}^{r} K_i = K$ and no K_i, $1 \leqq i \leqq r$, contains more than $[n/r]$ states.‡ Let $\mathcal{S} = \{F\} \cup \{G \mid G \subseteq K_i \text{ for some } i\}$. To see that $\mathcal{S}$ is an M_r-synthesis of A, we have merely to observe that any subset G of K can be expressed as $\bigcup_{i=1}^{r} G_i$, where $G_i = G \cap K_i \subseteq K_i$ for all i. Thus, for any H in $\mathcal{S}$, $H/0$ and $H/1$ are both the union of r elements of $\mathcal{S}$.

The size of $\mathcal{S}$ is no greater than $1 + r\,(2^{[n/r]} - 1)$, which is almost $r2^{[n/r]}$. We thus have:

Theorem 1: If A *is an* n *state sequential machine, with a single binary input, then there is an* M_r*-synthesis of* A *using at most* $r2^{[n/r]}$ *copies of* M_r.

Notice that Theorem 1 is not dependent upon the assumption that A has a single binary input. The machine A in that theorem can have any number of binary inputs. Of course, the appropriate generalization of the input module M_r, as given in Figure 3, must be used.

† Φ denotes the empty set.
‡ We use $[x]$ for "the smallest integer equal to or greater than x."

Example: We use a technique suggested by Theorem 1 to design a network for the sequential machine of Table I with final states {4, 5, 6}. We generate subsets in a sequential manner, and terminate when no new sets are required. Let $r = 2$ and let the states be divided into two sets K_1 {1, 3, 5} and $K_2 = \{2, 4, 6\}$. Now $\{4, 5, 6\}/0 = \{2, 4, 5\}$ and $\{4, 5, 6\}/1 = \{1, 2, 3, 4, 6\}$. If we intersect $\{2, 4, 5\}$ and $\{1, 2, 3, 4, 6\}$ each with K_1 and K_2, the inputs to the module realizing {4, 5, 6} must be connected to modules realizing {5}, {2, 4}, {2, 4, 6} and {1, 3}. The two derivatives of each of these sets are found among {2, 4, 5}, {6}, {1, 3}, {3}, {2, 5, 6}, {1, 2, 4, 5, 6} and Φ. Intersecting each of these sets with K_1 and K_2, we find that the network needs modules realizing the events {6}, {3}, {2, 6} and {1, 5}, in addition to the modules already used. Proceeding in this way, we find that the entire network also requires modules realizing {1}, {4} and {3, 5}. The completed network is shown in Fig. 6. Modules are labeled with the event (set of states) they realize. Those modules realizing a set including state 1, the start state, must initially give a 1 output. Inputs to the module are shown in no particular order, and inputs not shown are connected to 0.

The second bound uses the concept of partitions on the set of states of a finite automaton.[9] A *partition* on a set of states K is a set of disjoint, nonempty sets, called *blocks* whose union is K. If $A = (K, \{0, 1\}, \delta, q_0, F)$ is a sequential machine, we can associate with every string w in $\{0, 1\}^*$ a partition Π_w as follows:

(*i*) $\Pi_\epsilon = (\{q_1\}, \{q_2\}, \cdots, \{q_m\})$, where $K = \{q_1, q_2, \cdots, q_m\}$.†

(*ii*) For any w in $\{0, 1\}^*$, let Π_w be $(K_1, K_2, \cdots, K_r)$. Let Π_{w0} be the list of nonempty sets G such that $G = K_i/0$ for some i and Π_{w1} be the list of nonempty sets H such that $H = K_i/1$ for some i.

Example: Consider the machine of Table I. $\Pi_\epsilon = 1, 2, 3, 4, 5, 6)$ Π_0 is the list of sets of states that map to a single state under a 0 input. Thus, $\Pi_0 = (1, 245, 3, 6)$. Similarly, $\Pi_1 = (14, 26, 3, 5)$. Proceeding, we can calculate Π_{00} and Π_{01} from Π_0 by seeing which sets of states map onto a single block of Π_0 under inputs 0 and 1, respectively. For example, states 2, 4, 5 and 6 are those which map under a 0 input to one of the states 2, 4 or 5. We find $\Pi_{00} = (1, 2456, 3)$ and $\Pi_{01} = (14, 2356)$. Also, $\Pi_{10} = (1, 245, 3, 6)$ and $\Pi_{11} = (145, 26, 3)$.

A partition Π is said to *represent* a family of sets, namely those sets

† We denote partitions by lists of the blocks. Sometimes it is simpler to represent each block by a string of states not surrounded by brackets. Thus $(\{q_1, q_2\}, \{q_3\})$ will appear as (q_1q_2, q_3).

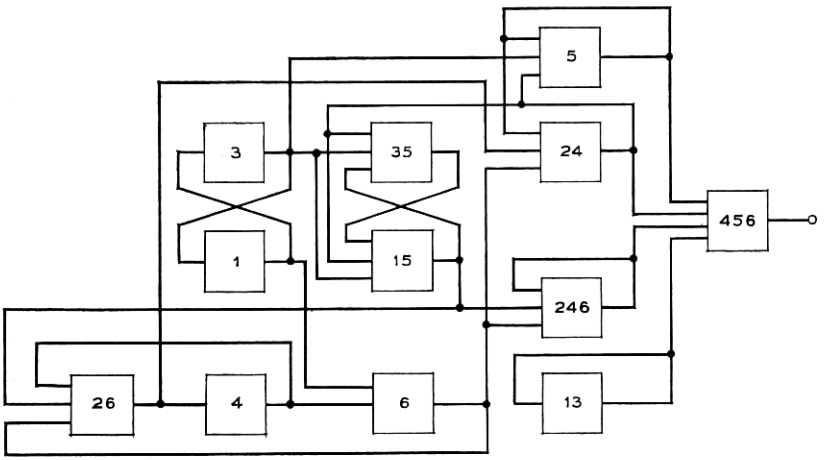

Fig. 6 — Network suggested by Theorem 1.

which are the union of some of the blocks of Π. For example, Π_{01} above represents the sets Φ, $\{1, 4\}$, $\{2, 3, 5, 6\}$ and $\{1, 2, 3, 4, 5, 6\}$. Suppose $\Pi_w = (K_1, K_2, \cdots, K_m)$ and G is the union of j of $K_1, K_2, \cdots, K_m$, say $G = K_{i_1} \cup K_{i_2} \cup \cdots \cup K_{i_j}$. Then for $a = 0$ or 1, $G/a = K_{i_1}/a \cup K_{i_2}/a \cup \cdots \cup K_{i_j}/a$ is represented by Π_{wa} and is, in fact, the union of, at most, j blocks of Π_{wa}. Armed with this observation, we prove:

Theorem 2: *For every* n *state sequential machine with single binary input* $A = (K, \{0, 1\}, \delta, q_0, F)$ *and* $r \geqq 2$, *there is an* M_r*-synthesis of* A *of size at most* $\frac{2r}{2r-1}(n^{1+\log_{r^2}} + 4n^{1+\log_{r^4}})$.

Proof: Let $j = [\log_r n]$. Define the blocks of those partitions Π_w, such that $|w| \leqq j$† to be *basic events*. We will choose $\mathfrak{I}$ to be a set of pairs (G, w), where $G \subseteq K$, w is in $\{0, 1\}^*$, and for each (G, w) in $\mathfrak{I}$, G is represented by Π_w. After constructing $\mathfrak{I}$, we construct $\mathfrak{S}$, an M_r-synthesis of A, from $\mathfrak{I}$ by $\mathfrak{S} = \{\Phi\} \cup \{G \mid (G, w) \text{ is in } \mathfrak{I} \text{ for some } w\}$. We construct $\mathfrak{I}$ by:

(*i*) $(F, \epsilon$ is in $\mathfrak{I})$.

(*ii*) If G is a basic event, then (G, ϵ) is in $\mathfrak{I}$.

(*iii*) Let (G, w) be in $\mathfrak{I}$, $|w| < j$, and let G be the union of k blocks of Π_w. We may choose $H_1, H_2, \cdots, H_r$ such that their union is $G/0$

† $|w|$ denotes the length of w.

and for each i, H_i is the union of from zero to $[k/r]$ blocks of Π_{w0}. Also, choose $J_1, J_2, \cdots, J_r$ such that their union is $G/1$ and for each i, J_i is the union of from zero to $[k/r]$ blocks of Π_{w1}. If H_i is not Φ or a basic event, add $(H_i, w0)$ to $\mathfrak{I}$. If J_i is not Φ or a basic event, add $(J_i, w1)$ to $\mathfrak{I}$.

We say that each $(H_i, w0)$ or $(J_i, w1)$ in $\mathfrak{I}$ is in the *family* of (G, w). We extend the notion of a family by saying that (G, w) is in its own family and if (H, x) is in the family of (G, w) and (J, y) is in the family of (H, x), then (J, y) is in the family of (G, w). The family of (G, ϵ), where G is F or a basic event, can be thought of as the set of elements that must be in $\mathfrak{I}$ because (G, ϵ) is in $\mathfrak{I}$.

We must show that $\mathfrak{S}$ is an M_r-synthesis of A. If (G, w) is in $\mathfrak{I}$, then G consists of at most $r^{i-|w|}$ blocks of Π_w. (Since $r^i \geqq n$, we have $r^{i-1} = [r^i/r] \geqq [n/r]$; $r^{i-2} = [r^{i-1}/r] \geqq [[n/r]/r]$ and so on.) We may conclude that if $|w| = j$, then G would be a basic event, and hence, for no G and w of length j is (G, w) in $\mathfrak{I}$. If G is a basic event or F, one can, by rule (*iii*) find $H_1, H_2, \cdots, H_r$ and $J_1, J_2, \cdots, J_r$ such that

$$G/0 = \bigcup_{i=1}^{r} H_i, \qquad G/1 = \bigcup_{i=1}^{r} J_i.$$

For all i, either $(H_i, 0)$ is in $\mathfrak{I}$ or $H_i = \Phi$ or H_i is a basic event, and either $(J_i, 1)$ is in $\mathfrak{I}$ or $J_i = \Phi$ or J_i is a basic event. In any case, all of $H_1, H_2, \cdots, H_r$ and $J_1, J_2, \cdots, J_r$ are in $\mathfrak{S}$. If G is in $\mathfrak{S}$ but G is neither a basic event not F, then it must be that (G, w) is in $\mathfrak{I}$ and $|w| < j$. But in this case, it again follows immediately from rule (*iii*) that $H_1, H_2, \cdots, H_r$ and $J_1, J_2, \cdots, J_r$ in $\mathfrak{S}$ can be found with $\bigcup_{i=1}^{r} H_i = G/0$ and $\bigcup_{i=1}^{r} J_i = G/1$.

We must now put a bound on the size of $\mathfrak{S}$. We do so by bounding the number of elements in the families of all (G, ϵ) in $\mathfrak{I}$. The sum of the sizes of all these families bounds the size of $\mathfrak{S}$.

Suppose G consists of k states and $m = [\log_r k]$. For each $i \geqq 0$, there are at most $(2r)^i$ elements (H, w) in the family of (G, ϵ) such that $|w| = i$. If (H, w) is in the family of (G, ϵ), then H consists of at most $r^{m-|w|}$ blocks of Π_w. Thus the family of (G, ϵ) contains no pair (H, w) such that $|w| \geqq m$. An upper bound on the size of the family of (G, ϵ) is $1 + 2r + (2r)^2 + \cdots + (2r)^{m-1}$. This number does not exceed $(2r)^m/(2r - 1)$. But $m \leqq 1 + \log_r k$, so $(2r)^{m-1} \leqq k^{1+\log_r 2}$.

We may conclude that the family of (F, ϵ) consists of at most $\frac{2r}{2r-1} n^{1+\log_r 2}$ elements. We must also bound the families of the basic events, and do so by the following argument.

Let N_k be the number of basic events consisting of exactly k states. There are $1 + 2 + 4 + \cdots + 2^j$ partitions Π_w where $|w| \leqq j$. The number of these partitions is at most 2^{j+1}; the blocks of each partition have among them a total of n states. Thus:

$$\sum_{k=1}^{n} kN_k \leqq n2^{j+1}. \tag{1}$$

An upper bound on the sum of the sizes of the families of all the basic events is

$$\sum_{k=1}^{n} N_k \frac{2r}{2r-1} k^{1+\log_r 2}.$$

Since k does not exceed n in the summation, we have

$$\sum_{k=1}^{n} N_k \frac{2r}{2r-1} k^{1+\log_r 2} \leqq \frac{2r}{2r-1} n^{\log_r 2} \sum_{k=1}^{n} kN_k .$$

Using equation (1), we see that the sum of the sizes of the families of all basic events is bounded above by $2r/(2r-1)n^{1+\log_r 2}2^{j+1}$. Since $j \leqq 1 + \log_r n$, this bound becomes $8r/(2r-1)n^{1+\log_r 4}$.

Including the family of (F, ϵ), we see that the size of $\mathcal{S}$ is no greater than $\frac{2r}{2r-1}(n^{1+\log_r 2} + 4n^{1+\log_r 4})$.

We comment that a straightforward generalization of this argument shows that every sequential machine with p binary inputs (2^p symbol input alphabet) can be realized by a network of at most $2^p r/(2^p r - 1) \cdot (n^{1+p\log_r 2} + 4^p n^{1+p\log_r 4})$ copies of the generalization of the module M_r. Thus, for any number of binary inputs p, and any $c > 0$, there are constants r and k such that any n state sequential machine with p binary inputs can be realized by a network of at most kn^{1+c} copies of a module with $2^p r$ intermodule leads.

Example: Theorem 2 suggests the design of a network of copies of M_2 for the machine of Table I with states 4, 5 and 6 final. That machine has 6 states and $[\log_2 6] = 3$. However, in this case the construction of $\mathcal{S}$ given in Theorem 2 will not require the addition of any pair (G, w) where $|w| > 1$. So we may restrict ourselves to consideration of certain sets represented by the partitions Π_w, for $|w| \leqq 2$. These were calculated in the previous example:

$$\Pi_\epsilon = (1, 2, 3, 4, 5, 6) \qquad \Pi_{00} = (1, 2456, 3)$$

$$\Pi_0 = (1, 245, 3, 6) \qquad \Pi_{01} = (14, 2356)$$

$$\Pi_1 = (14, 26, 3, 5) \qquad \Pi_{10} = (1, 245, 3, 6)$$
$$\Pi_{11} = (145, 26, 3).$$

We begin by placing $(\{4, 5, 6\}, \epsilon)$ in $\mathfrak{I}$. $\{4, 5, 6\}/0$ is the basic event $\{2, 4, 5\}$, and $\{4, 5, 6\}/1$ is the union of three basic events $\{2, 6\}$, $\{3\}$ and $\{1, 4\}$. These three must be formed into two groups; we choose to realize $\{2, 3, 6\}$ and $\{1, 4\}$. We place $(\{2, 4, 5\}, \epsilon)$ and $(\{1, 4\}, \epsilon)$ in $\mathfrak{I}$, since these are basic events, but since $\{2, 3, 6\}$ is not a basic event, we place $(\{2, 3, 6\}, 1)$ in $\mathfrak{I}$.

$\{2, 4, 5\}/0 = \{2, 4, 5\} \cup \{6\}$, so $(\{6\}, \epsilon)$ is placed in $\mathfrak{I}$. $\{2, 4, 5\}/1$ can be expressed as $\{2, 3, 6\} \cup \{5\}$. We thus place $(\{5\}, \epsilon)$ in $\mathfrak{I}$. $\{1, 4\}/0 = \{3\}$ and $\{1, 4\}/1 = \{2, 6\}$. Each of these are basic events, so $(\{3\}, \epsilon)$ and $(\{2, 6\}, \epsilon)$ are placed in $\mathfrak{I}$. $\{2, 3, 6\}/0 = \{1\} \cup \{6\}$. These are basic events, so we add $(\{1\}, \epsilon)$ to $\mathfrak{I}$. $\{2, 3, 6\}/1 = \{1, 4\} \cup \{5\}$; these basic events are each represented in $\mathfrak{I}$ already. Proceeding, we find that the basic events added to $\mathfrak{I}$ require no new events, basic or not. The resulting network is shown in Fig. 7.

IV. CONCLUSIONS

We have considered the design of synchronous sequential machines by networks of a fixed module. This design has various advantages, including speed and ease of production using batch fabrication. It was shown that there is a family of modules M_r, $r \geqq 1$, such that any n state sequential machine with a single binary input can be realized by a network of at most p copies of M_r, when p is the minimum of $r2^{\lceil n/r \rceil}$ and $\frac{2r}{2r-1}(n^{1+\log_r 2} + 4n^{1+\log_r 4})$.

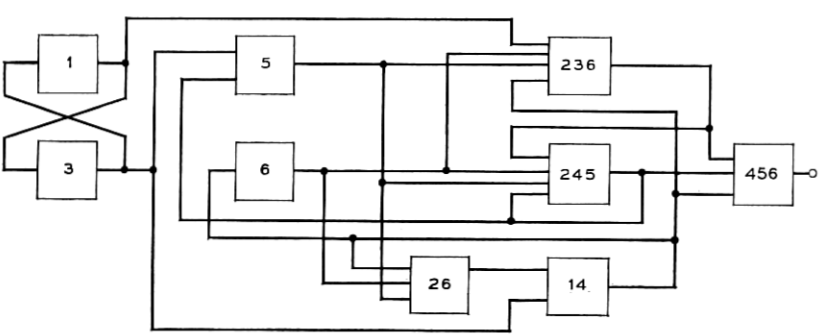

Fig. 7 — Network suggested by Theorem 2.

We feel that the type of design suggested in this paper leads to many interesting questions. In particular, the bounds expressed in Theorems 1 and 2 do not seem to be attained, or even approximated, in most cases. Efficient search techniques will probably yield much better networks than indicated; there is every reason to suspect that the bounds themselves can be improved, even if we restrict consideration to isomorphic networks.

REFERENCES

1. Weiner, P., and Hopcroft, J. E., "Modular Decomposition of Synchronous Sequential Machines," Proc. IEEE 8th Annual Symp. on Switching and Automata Theory, Austin, Texas, October 1967, pp. 233–239.
2. Hsieh, E. P., Tan, C. J., and Newborn, M. M., "Uniform Modular Realization of Sequential Machines," Proceedings of 1968 Association for Computing Machinery Conference, Las Vegas, Nevada, August 1968, pp. 613–621.
3. Weiner, P. and Hopcroft, J. E., "Bounded Fan-in, Bounded Fan-out Uniform Decompositions of Synchronous Sequential Machines," Proc. IEEE, *46,* No. 4 (July 1968), pp. 1219–1220.
4. Arnold, T. F., Tan, C. J., and Newborn, M. M., "Iteratively Realized Sequential Circuits," Proc. IEEE 9th Annual Symp. on Switching and Automata Theory, Schenectady, N. Y., October 1968, pp. 431–448.
5. Ullman, J. D. and Weiner, P., "Universal Two State Machines: Characterization Theorems and Decomposition Schemes," Proc. IEEE 9th Annual Symp. on Switching and Automata Theory, Schenectady, N.Y., October 1968, pp. 413–426.
6. Curtis, H. A., "Polylinear Sequential Circuit Realizations of Finite Automata," IEEE Transactions on Computers, *C-17,* No. 3 (March 1968) pp. 251–258.
7. Brzozowski, J. A., "Derivatives of Regular Expressions," Journal of Association for Computing Machinery *11,* No. 4 (October 1964), pp. 481–494.
8. Elgot, C. C. and Rutledge, J. D., "Operations on Finite Automata," Proc. IEEE 2nd Annual Symp. on Switching Theory and Logical Design, October 1961, pp. 129–132.
9. Hartmanis, J. and Stearns, R. E., Algebraic Structure Theory of Sequential Machines, Englewood Cliffs, N. J.: Prentice Hall, 1966.