memory to store the longer sequences of instructions that it requires.

Synthesizing SMs in the forms suggested by the results of this paper does not appear to be practical primarily because of the long sequences required by a universal machine to realize any arbitrary machine. For example, it has been shown that for an r-state universal machine of the type shown in Fig. 4(a), the longest sequence required by an arbitrary r-state SM grows exponentially with r (see reference [3], pp. 94–97).

ACKNOWLEDGMENT

The author wishes to thank Prof. A. K. Susskind and Dr. R. L. Martin of M.I.T. for stimulation during the course of this research and for constructive comments on and review of preliminary manuscripts. He also wishes to thank one of the referees for some useful suggestions.

REFERENCES


Unitary Shift-Register Realizations of Sequential Machines

C. C. SU, MEMBER, IEEE, AND S. S. YAU, MEMBER, IEEE

Abstract—The problem of determining secondary state assignments for sequential machines such that the binary memory elements are connected in the form of shift registers is studied. An algorithm for finding such state assignments is developed. One or more code words may be assigned to a state of the sequential machine. The only restriction is that the realizations be unitary. A single shift-register realization of a sequential machine is unitary if and only if all the code words assigned to a state have the same first digits. A multiple shift-register realization of a sequential machine is unitary if and only if all the code words assigned to a state have l identical digits, where l is the number of shift registers in the realization. With our technique, the unitary realizations with the minimum number of shift registers can be obtained for any finite, deterministic, synchronous, and reduced (minimal-state) sequential machine, each of whose states has a nonempty predecessor set. The algorithm is suitable for programming on digital computers.

Index Terms—Algorithms, many-to-one state assignments, sequential machines, shift-register realizations, unitary coding.

Manuscript received February 16, 1967; revised December 11, 1967. This work was supported in part by the U. S. Air Force Office of Scientific Research under Grants AF-AFOSR-98-65 and AF-AFOSR-1292-67, and in part by the Army, Navy, and Air Force Joint Services Electronics Program under Contract N00014-66-C0020-A03 (Identification Number NR 373-502/3-14-66 Electronics Branch).

C. C. Su was formerly with the Information-Processing and Control Systems Laboratory and the Department of Electrical Engineering, Northwestern University, Evanston, Ill. He is now with the Burroughs Corporation, Detroit, Mich. S. S. Yau is with the Information-Processing and Control Systems Laboratory and the Department of Electrical Engineering, Northwestern University, Evanston, Ill.
realizations. In this paper, a special type of many-to-one state assignment technique for shift-register realization will be developed. The state assignment obtained by this method is called “unitary state assignment.”

The formal definitions of unitary state assignments and other related terms will be given in the next section. In Section III we shall present the unitary state assignment technique for single shift-register realization, and in Section IV for multiple shift-register realization. This technique can be applied to completely specified as well as incompletely specified, synchronous, deterministic, sequential machines with every state having at least a predecessor. [8]

II. Definitions and Notations

A "sequential machine" $M$ may be considered as a 5-tuple $(I, W, S, f, g)$, where $I = \{i_1, i_2, \ldots, i_r\}$, $W = \{w_1, w_2, \ldots, w_s\}$, $S = \{s_1, s_2, \ldots, s_t\}$ are finite sets called the “input set,” the “output set,” and the “state set,” respectively; and $f: S \times I \rightarrow S$, and $g: S \rightarrow W$ (Moore model) or $S \times I \rightarrow W$ (Mealy model) are two mappings, called the "next-state function" and the “output function,” respectively, where $S \times I$ is the Cartesian product of $S$ and $I$. $M$ is “deterministic” if both $f$ and $g$ are single-valued at every point of their domains; $M$ is “synchronous” if every state transition (and hence every output change) is synchronized with the signal from the central synchronizing clock. [9] $M$ is “completely specified” if both $f$ and $g$ are defined (or specified) at every point of their domains; otherwise, $M$ is said to be “incompletely specified.” The next-state function $f$ and the output function $g$ are usually expressed in the form of a table, called the “flow table,” which consists of two parts, namely, “state transition table” and “output table,” describing $f$ and $g$, respectively. Since we are interested in the shift-register realizations of sequential machines which involve only the state behavior or the next-state function $f$ of a sequential machine, the output function $g$ will not be considered in this paper.

The next-state function $f$ of a sequential machine $M$ of $r$ states can be partially described by the so-called “skeleton matrix” $K(M) = [k_{ij}]$, which is a matrix of order $r$ whose entry $k_{ij}$ is 1 if there exists an input $i_k$ such that $s_j = f(s_i, i_k)$ and is 0 otherwise. [8] It is obvious that different sequential machines may have the same skeleton matrix. We shall use $G[K(M)]$ to denote the set of all sequential machines with the same skeleton matrix $K(M)$ after properly relabeling their states. It will be seen that $G[K(M)]$ plays an important role in the shift-register realizations of sequential machines.

Let $B$ be a subset of $S$. The successor set $f(B)$ and the predecessor set $f^{-1}(B)$ of $B$ are defined as follows.

\[
f(B) = \{ s_i : s_i = f(s_j, i_k) \text{ for some } s_j \in B \text{ and } i_k \in I \} \tag{1}
\]
\[
f^{-1}(B) = \{ s_i : s_j = f(s_i, i_k) \text{ for some } s_j \in B \text{ and } i_k \in I \}. \tag{2}
\]

For the empty subset $\phi$ of $S$, we define $f(\phi) = f^{-1}(\phi) = \phi$.

A “binary code” $C_m$ is the set of all possible sequences of $m$ binary digits, where $m$ is a positive integer, called the “length” of $C_m$. Each sequence in $C_m$, denoted by $\alpha \in C_m$, is called a “code word” of $C_m$. A “coding” on a state set $S = \{s_1, s_2, \ldots, s_t\}$ is an assignment of one or more code words to each element of $S$ such that each code word is assigned to at most one element of $S$.

Definition 1: A coding $\text{C}_m$ on a state set $S$ is “unitary” if for each $s \in S$, all code words assigned to $s$ have identical first digits.

A “partition” on a set $S$ is a collection of disjoint subsets $\{A_1, A_2, \ldots, A_n\}$ of $S$, called “blocks,” such that the union of $A_1, A_2, \ldots, A_n$ is $S$. A “binary partition” is a partition which consists of exactly two nonempty blocks. A “null partition” is a partition in which every block consists of exactly one element of $S$.

A “set system” on a set $S$ is a collection of subsets $\{A_1, A_2, \ldots, A_n\}$ of $S$, also called “blocks” but not necessarily disjoint, such that each element of $S$ is in at least one subset. [10] A “binary set system” is a set system consisting of exactly two nonempty blocks. A “null set system” is a set system in which every block consists of exactly one element of $S$. Hence, a partition is always a set system, but a set system may or may not be a partition. Furthermore, the null partition of a set is unique, but its null set systems are not unique.

Let $\theta$ and $\theta'$ be two set systems (or two partitions). We write $\theta \supseteq \theta'$ if for each block $A' \in \theta'$ there exists a block $A \in \theta$ such that $A' \subseteq A$. Let $P$ and $P'$ be two partitions. We write $P = P'$ if and only if $P \supseteq P'$ and $P' \supseteq P$ if $P \supseteq P'$ but $P \neq P'$. When we say “$\theta$ is a set system (or partition),” we mean “$\theta$ is a set system (or partition) on the state set $S$ of a sequential machine $M$,” unless otherwise specified. For a coding $C_2$ on a set system $\theta$, each code word $\alpha \in C_2$ which is not assigned to any block of $\theta$ is assigned to a null subset $\phi$.

Definition 2: A “transition graph” $T_{s}^{\theta}$ of a set system $\theta$ with a coding $C_2$ is a graph of $2^t$ vertices and a number of directed branches, in which each vertex is labeled with a distinct code word of $C_2$, and for each branch directed from a vertex labeled $\alpha$ to a vertex labeled $\alpha'$ there exist a state $s$ in a block $A$ of $\theta$ coded with $\alpha$ and a state $s'$ in a block $A'$ of $\theta$ coded with $\alpha'$, such that $s' = f(s, i_j)$ for some input $i_j$. $T_{s}^{\theta}$ is said to be “complete” if $f(s, i_j) = s'$ implies that there is at least one branch directed from each of the vertices corresponding to the blocks of $\theta$ containing $s$ to at least one of the vertices corresponding to the blocks of $\theta$ containing $s'$; otherwise, $T_{s}^{\theta}$ is said to be “incomplete.”

It is noted that each vertex of $T_{s}^{\theta}$ corresponds to exactly one block (may be $\phi$) of $\theta$ because each $\alpha \in C_2$ is assigned to one block (or $\phi$) of $\theta$. Furthermore, the transition graph $T_{s}^{\theta}$ of $\theta$ with a coding $C_2$ is not unique. For convenience, we shall use $T_{s}^{\theta}$ or simply $T$ to denote $T_{s}^{\theta}$ whenever no confusion exists. When the block $A$ of $\theta$ is assigned to a vertex of $T_{s}^{\theta}$, $A$ may be employed to represent both the block of $\theta$ and the vertex of $T_{s}^{\theta}$. 
For $\alpha \in C_k$ and $A \in T_\beta$, mappings $A : C_k \rightarrow T_\beta$ and $C : T_\beta \rightarrow C_k$ are defined as follows.

$A(\alpha) = A'$, the block to which $\alpha$ is assigned, \hspace{1cm} (3)

$C(A) = \alpha'$, the code word which is assigned to $A$. \hspace{1cm} (4)

Let $\{a_1, a_2, \ldots, a_p\}$ be a subset of $\{1, 2, \ldots, k\}$, where $p$ is a positive integer smaller than $k$. The mapping $D_{a_1, a_2, \ldots, a_p(\alpha)}$, defined as follows for $\alpha \in C_k$:

$$D_{a_1, a_2, \ldots, a_p(\alpha)} = \beta,$$

where $\beta \in C_{k-p}$ is formed by deleting $a_1$th, $a_2$th, $\ldots$, $a_p$th digits from $\alpha$. \hspace{1cm} (5)

The above definitions may be illustrated by the following example. Consider a sequential machine $M_1$ whose state transition table is shown in Fig. 1(a). $\theta_1 = \{A_1, A_2, A_3\} = \{(s_1, s_2), (s_3, s_4), (s_5, s_6)\}$ is a set system, one of whose complete transition graphs with a certain coding $C_3$ is shown in Fig. 1(b). $\theta_2 = \{(s_1, s_2), (s_3, s_4), (s_5, s_6)\}$ is a partition (hence also a set system), whose complete transition graph with a certain coding $C_2$ is shown in Fig. 1(c). For $\theta_1$ with the coding $C_3$ shown in Fig. 1(b), we have

$$A(101) = (s_1, s_2), \hspace{1cm} A(011) = \phi,$$

$$C(s_1, s_2) = 001, \hspace{1cm} C(s_3, s_4) = 010,$$

$$D_2(010) = 00, \hspace{1cm} D_3(001) = 1,$$

$$D_3[C(s_1, s_2)] = D_3(101) = 10.$$

**Definition 3:** For each positive integer $k$, $R_k$ is a graph consisting of $2^k$ vertices in which each vertex is labeled with one distinct code word $\alpha \in C_k$, and there is a branch directed from $A(\alpha)$ to $A(\alpha')$ if and only if $D_p(\alpha) = D_p(\alpha')$.

It is noted that $R_k$ is actually the transition graph of a shift register of length $k$ receiving unrestricted binary input. This type of graph is usually called Good's diagram. \hspace{1cm} (6)

For instance, $R_2$ and $R_3$ are shown in Fig. 2.

**III. Single Shift-Register Realizations of Sequential Machines**

In this section, single shift-register realizations of sequential machines with unitary state assignments will be discussed. In this case, all the code words assigned to the same state must have the identical first digits. We shall give a necessary and sufficient condition for a sequential machine to be unitarily realizable with a single shift register, and then present a technique for such a realization.

Before considering the unitary state assignments, let us discuss single shift-register realizations for sequential machines with the same skeleton matrix.

**Theorem 1:** If a sequential machine $M$ with a coding $C_m$ on $S$ has a single shift-register realization, then each sequential machine $M' \in G[K(M)]$ with the same coding $C_m$ has a single shift-register realization.

**Proof:** Let each state of $M$ be coded with at least one code word $\alpha \in C_m$. Let $y_j$, $j = 1, 2, \ldots, m$, be the state variable corresponding to the $j$th digit of $\alpha \in C_m$, and $Y_j$ be the switching function for $y_j$. Since $M$ has a single shift-register realization with the coding $C_m$, we have

$$Y_j = y_{j-1}, \hspace{1cm} j = 2, 3, \ldots, m.$$  \hspace{1cm} (6)

(6) implies that after we code all the next-states in the state transition table of $M$ with $C_m$, the columns corresponding to each $Y_j$, $j = 2, 3, \ldots, m$, under all inputs are identical, or can be made identical by assigning proper code words to the unspecified entries. In other words, if $s_i \in f(s_k)$ and $s_i$ is assigned a code word $\alpha$, then
each member of \( f(s_k) \) is assigned at least one code word which has the same jth digit, \( j = 2, 3, \ldots, m \), as that of \( \alpha \). Now, let us consider a sequential machine \( M' \in G[K(M)] \), where \( K(M) \) is the skeleton matrix of \( M \) with the same coding \( C_m \). Let \( f' \) be the next-state function of \( M' \). Since \( M' \in G[K(M)] \), the set \( f'(s_k) \) is identical to \( f(s_k) \), \( k = 1, 2, \ldots, r \). Hence, with the same coding \( C_m \), the jth digit, \( j = 2, 3, \ldots, m \) of each entry in the coded state transition table of \( M' \) is identical to the jth digit of its corresponding entry in the coded state transition table of \( M \). Consequently, the switching function \( Y_j' \) for the state variable \( y_j \) of \( M' \), \( j = 2, 3, \ldots, m \), will be same as \( Y_j \) of \( M \) which is given by (6). Therefore, \( M' \) has a single shift-register realization with the coding \( C_m \), and this completes the proof of the theorem.

It follows from Theorem 1 that while the existence or nonexistence of transitions among the states of a sequential machine \( M \) is vital to whether \( M \) can have a single shift-register realization, the inputs causing the state transitions are irrelevant. Furthermore, it is obvious that if any \( f(s_k) \) of \( M \) consists of more than two different states, \( M \) cannot be realizable by using a single shift register. Hence, the sequential machines considered in this section are assumed to have at most two different states in every \( f(s_k) \).

**Definition 4:** A transition graph \( T \) of a set system \( \theta \) with a coding \( C_k \) is a “subgraph” of \( R_k \) if each branch of \( T \) is a branch of \( R_k \) after the vertices of \( T \) and the vertices of \( R_k \) with the same code words are identified.

**Definition 5:** A set system \( \theta \) is “unitarily embeddable in a shift register of length \( k^n \)” (“k-SRUE”) if there exists a unitary coding \( C_k \) on \( \theta \) such that some complete transition graph \( T^* \) of \( \theta \) is a subgraph of \( R_k \). \( \theta \) is “SRUE” if \( \theta \) is k-SRUE for some positive integer \( k \).

**Definition 6:** Let \( \theta = \{A_1, A_2, \ldots, A_s \} \) be a set system and \( T \) a transition graph of \( \theta \) with a coding \( C_k \). The “expansion” \( E(T) \) of \( \theta \) is a graph consisting of \( 2k+1 \) vertices, each labeled with one \( \alpha \in C_{k+1} \). To the vertex labeled with \( \alpha \) is assigned a subset \( B(\alpha) \) of \( S \), where

\[
B(\alpha) = A[D_{k+1}(\alpha)] \cap f[A[D_1(\alpha)]].
\]

There is a branch directed from \( B(\alpha) \) to \( B(\alpha') \) if \( D_{k+1}(\alpha') = D_1(\alpha') \) and if there exist \( s_1 \in B(\alpha) \) and \( s_2 \in B(\alpha') \) such that \( s_2 = f(s_1, i_j) \) for some input \( i_j \).

It is noted that \( E(T) \) is always a subgraph of \( R_{k+1} \). For any positive integer \( p \), \( E^p(T) \) is defined as follows:

\[
E^p(T) = E[E^{p-1}(T)] = \cdots = E \cdots E(T);
\]

that is, the expansion \( E \) operates on \( T \) \( p \) times. Let \( \theta \) and \( \theta' \) be the sets of all subsets of \( S \) assigned to the vertices of \( T \) and \( E^p(T) \), respectively. Then, \( \theta' \) is called the \( "p\)-expansion” of \( \theta \), denoted by \( E^p(\theta) \). It follows from (7) that the expansion of a set system cannot be a set system unless each state has a nonempty predecessor set.

Consider the sequential machine \( M_2 \) whose state transition table is shown in Fig. 3(a). The set system \( \theta = \{ (s_1), (s_2), (s_1, s_2), (s_4, s_6) \} \) is 2-SRUE because there exists a unitary coding \( C_2 \) on \( \theta \) such that a complete \( T^*_2 \) shown in Fig. 3(b) is a subgraph of \( R_2 \). The expansion \( E(T^*_2) \) is shown in Fig. 3(c), where the dotted branches are the branches in \( R_2 \) but not in \( E(T^*_2) \). The expansion \( E(\theta) \) is \( \{ (s_1), (s_2), (s_1, s_2), (s_4, s_6) \} \), which is also a set system.

Throughout the remaining part of this paper it will be assumed that the variables \( \delta, \xi, \gamma \) are binary; i.e., each of them can only assume the value 0 or 1. \( \bar{\delta}, \bar{\xi}, \bar{\gamma} \) represent the complements of \( \delta, \xi, \gamma \), respectively. For instance, \( \bar{\delta} = 0 \) if \( \delta = 1 \), and \( \bar{\xi} = 1 \) if \( \delta = 0 \).

**Theorem 2:** If \( \theta \) is a k-SRUE set system of a sequential machine \( M \), then there exists a binary partition \( P = \{ B_0, B_1 \} \) such that \( \theta = E^{k-1}(P) \).

**Proof:** Let \( \theta = \{ A_1, A_2, \ldots, A_s \} \). For each \( \alpha \in C_{k-1} \), let

\[
B(\alpha) = A(\alpha\delta) \cup A(\alpha\bar{\delta}).
\]

Then we have

\[
\bigcup_{\alpha \in C_{k-1}} B(\alpha) = \bigcup_{A_i \in \theta} A_i = S.
\]
Since $\theta$ is $k$-SRUE, we obtain
\begin{align}
 f[A(\alpha\delta)] &\subseteq A(\xi\alpha) \cup A(\xi\alpha) \supseteq f[A(\alpha\delta)] \quad (11) \\
 f[B(\alpha)] &= f[A(\alpha\delta) \cup A(\alpha\delta)] \\
 &= f[A(\alpha\delta)] \cup f[A(\alpha\delta)] \\
 &\subseteq A(\xi\alpha) \cup A(\xi\alpha) \\
 &\subseteq B[\xi D_k(\alpha)] \cup B[\xi D_{k-1}(\alpha)]. \quad (12)
\end{align}

Let $\theta_1 = \{ B(\alpha) | \alpha \in C_{k-1} \}$. Then, it follows from (10) that $\theta_1$ is a set system, and from (12) that $\theta_1$ is $(k-1)$-SRUE.

Now, assume $E(\theta_1) = \theta'' = \{ H_1, H_2, \cdots, H_p \}$. For each $\alpha \delta \in C_k$, if $s \in A(\alpha\delta)$, then
\begin{equation}
 s \in B(\alpha), \quad (13)
\end{equation}
and if $f^{-1}(s) \neq \phi,$
\begin{equation}
 f[B(D_k(\alpha\delta))] = f[A(D_k(\alpha\delta) \cup A[D_k(\alpha\delta)] \subseteq A(\alpha\delta) \supseteq \{ s \}. \quad (14)
\end{equation}

From Definition 6 and (13) and (14), we have
\begin{equation}
 H(\alpha\delta) = B(\alpha) \cap f[B(D_k(\alpha\delta))] \supseteq \{ s \}. \quad (15)
\end{equation}

It follows from (15) that
\begin{equation}
 H(\alpha\delta) \supseteq A(\alpha\delta) \quad \text{for each } \alpha \delta \in C_k. \quad (16)
\end{equation}

On the other hand, for each $\alpha \delta \in C_k$, if $s \in H(\alpha\delta)$, it follows from (15) that
\begin{equation}
 s \in B(\alpha) = A(\alpha\delta) \cup A(\alpha\tilde{\delta}), \quad (17)
\end{equation}
and
\begin{equation}
 s \in f[B(D_k(\alpha\delta))] = f[A(D_k(\alpha\delta))] \subseteq A[D_k(\alpha\delta)] \subseteq A(\alpha\tilde{\delta}). \quad (18)
\end{equation}

By comparing (17) and (18), and from the fact that the coding is unitary, we find that $s \in A(\alpha\delta)$. Therefore,
\begin{equation}
 A(\alpha\delta) \supseteq H(\alpha\delta) \quad \text{for each } \alpha \delta \in C_k. \quad (19)
\end{equation}

Because of (16) and (19), we have
\begin{equation}
 A(\alpha\delta) = H(\alpha\delta) \quad \text{for each } \alpha \delta \in C_k. \quad (20)
\end{equation}

Consequently, $\theta = \theta''$. Thus, we have shown that there exists a $(k-1)$-SRUE set system $\theta_1$ such that $\theta = E(\theta_1)$.

By induction on $k$ we can find a sequence of set systems $\theta_j$, $j = 1, 2, \cdots, k-1$, such that $\theta_j$ is $(k-j)$-SRUE and $\theta_j = E(\theta_{j+1})$, $j = 1, 2, \cdots, k-2$. Because $\theta_{k-1}$ is 1-SRUE, $\theta_{k-1}$ is a binary partition $\{ B_0, B_1 \}$ and $\theta = E^{k-1}(\theta_{k-1})$. This completes the proof of the Theorem.

The above proof indicates that if $\theta$ is a $k$-SRUE set system, the process of obtaining the $(k-1)$-SRUE set system $\theta_1 = \{ B(\alpha) | \alpha \in C_{k-1} \}$ from $\theta$ is precisely the inverse of the expansion of $\theta$. We shall call this process the "contraction" of $\theta$, denoted by $E^{-1}(\theta)$.

Let $P = \{ B_0, B_1 \}$ be a binary partition with $C(B_0) = 0$ and $C(B_1) = 1$. It is obvious that $T_{P}$ is a subgraph of $R_1$. Assume that $\alpha = \delta \gamma \in C_2$ and $s \in A(\alpha) \in E(P)$. Then, $s \in B_1$ because $s \in A(\alpha) = B_1 \cap f(B_1)$. In other words, a state in $B_1$ cannot appear in the blocks of $E(P)$ whose code words begin with $\delta$. By using similar argument, we can show that the statement is true for $E^p(P)$, where $p$ is any positive integer. Hence, we have the following lemma.

Lemma 1: Let $P = \{ B_0, B_1 \}$ be a binary partition with $C(B_i) = \delta, i = 0, 1$, and $s \in B_1$. Then, $s \in A(\alpha) \in E^{p-1}(P)$ only if $D_3, \ldots, p(\alpha) = \delta$, where $p$ is any integer not smaller than 2.

Theorem 3: Let $\theta$ be a SRUE set system. For any positive integer $p$, $\theta' = E^{p}(\theta)$ is a SRUE set system.

Proof: Assume $p=1$. Since $\theta$ is SRUE, there exists an integer $k$ such that $\theta$ is $k$-SRUE. Let $\theta = \{ A_1, A_2, \ldots, A_n \}$ be a $k$-SRUE set system with the coding $C_n$. Then, for any $s \in S$, there exists an $A(\alpha)$ such that $s \in A(\alpha)$, where $\alpha \in C_k$ and
\begin{equation}
 f^{-1}(s) \cap A[D_k(\alpha) \cup A[D_k(\alpha)]] \neq \phi. \quad (21)
\end{equation}

Let $E(\theta) = \theta' = \{ B_1, B_2, \cdots, B_m \}$, where $m = 2n$. It follows from Definition 6 that
\begin{equation}
 B(\alpha) = A[D_{k+1}(\alpha)] \cap f[A[D_k(\alpha)]]
 = A(\alpha) \cup f[A[D_k(\alpha)]] \quad (22)
\end{equation}
\begin{equation}
 B(\alpha) = A(\alpha) \cap f[A[D_k(\alpha)]] \quad (23)
\end{equation}
\begin{equation}
 B(\alpha) \cup B(\alpha) = A(\alpha) \cap f[A[D_k(\alpha)]] \cup A[D_k(\alpha)] \subseteq \{ s \}. \quad (24)
\end{equation}

Hence, $\theta'$ is a set system because it contains all elements of $S$. It follows from Theorem 2 that there exists a binary partition $P = \{ B_0, B_1 \}$ such that $\theta = E^{k-1}(P)$.

Now, assume that $s \in B(\alpha)$ for some $\alpha \in C_{k+1}$ and $f(s, i_j) = s'$ for some $s' \in B_1$ and some input $i_j$. Then, $s \in A[D_{k+1}(\alpha)]$ because of (22). Let $X_1$ be the set of states defined as follows:
\begin{equation}
 X_1 = \{ s | s \in B_1 \cap f(A[D_{k+1}(\alpha)]) \}. \quad (25)
\end{equation}

Then, $X_1 \subseteq A[D_k(b_{k+1}(\alpha))]$ by virtue of Lemma 1. Obviously, $s' \in X_1$. Furthermore,
\begin{equation}
 B(D_{k+1}(\alpha)) = A[D_{k+1}(\alpha)] \cap f[A[D_{k+1}(\alpha)]]
 = A[D_{k+1}(\alpha)] \cap f[A[D_{k+1}(\alpha)]]
 = X_1 \subseteq \{ s' \}. \quad (26)
\end{equation}

Thus, $\theta'$ is $(k+1)$-SRUE because $C_{k+1}$ is unitary and because the graph $T_{P'} = E(T_{\theta'})$ is complete and a subgraph of $R_{k+1}$. Hence, the proof for $p=1$ is completed. Assume that the theorem is true for $p=q$, i.e., $\theta' = E^{q}(\theta)$ is a SRUE set system. Then, $\theta'$ is $l$-SRUE for some positive integer $l$, and following the same line of argument, $E^{q}(\theta)$ is SRUE. It follows from (8) that $E^{q}(\theta) = E[E^{q}(\theta)] = E^{q+1}(\theta)$. Hence, the theorem is also true for $p=q+1$. The proof is thus completed by induction on $p$.

The following corollaries are direct consequences of the above theorems.
Corollary 1: If a set system \( \theta \) is \( k \)-SRUE, then
\[
E^{-1}E^{p+1}(\theta) = E^{-p+1}E^{-1}(\theta) = E^{-p}(\theta)
\]  
(27)
for any integer \( p \), \( 0 \leq p \leq k \), where \( E^0(\theta) \) is defined as \( \theta \).

Corollary 2: Let \( \theta \) be a \( k \)-SRUE set system and \( \theta' = E^p(\theta) \). Then, \( \theta' \) is a \((k+p)\)-SRUE set system for all integers \( p \geq -k \).

Definition 7: A binary partition \( P \) is “compatible” if for each \( s \in S \) distinct members of \( f(s) \) appear in different blocks of \( P \).

Let \( P = \{B_0, B_1\} \) be a compatible binary partition. If \( (s_o, s_b) \subseteq B_1 \), \( (s_t, s_g) \subseteq B_1 \), and \( f(s_o, i_1) = s_o, f(s_b, i_2) = s_b \) for some inputs \( i_1 \) and \( i_2 \), then we write \( s_o \rightarrow s_b \) or \( f_i(s_o) = s_b \), and \( s_t \rightarrow s_g \) or \( f_i(s_t) = s_g \). We may also denote the above relations by writing \( f^{-1}(s_o) \subseteq s_o \) and \( f^{-1}(s_b) \subseteq s_b \).

Definition 8: Let \( (s_a, s_b) \subseteq B_2 \) and \( f_i(s_a, s_b) = (s_c, s_d) \). \( (s_a, s_b) \) is “\( \xi \)-resolved” if \( s_c = s_d \), or if either \( s_c \) or \( s_d \) is unspecified. \( (s_a, s_b) \) is “resolved” if it is both \( 0 \)-resolved and \( 1 \)-resolved. \( (s_a, s_b) \) is “resolvable” if either \( (s_a, s_b) \) is resolved or both \( f_i(s_a, s_b) \) known to be resolvable.

A pair \( (s_a, s_b) \) is “unresolvable” if it is not resolvable.

\( \omega \) pairs \( (s_{o\alpha}, s_{b\beta}) \subseteq B_3 \), \( j = 1, 2, \ldots, \omega \), form an “unresolvable chain” if they have the property that
\[
(s_{o\alpha}, s_{b\beta}) \rightarrow (s_{o\alpha+1}, s_{b\beta+1}) \rightarrow \cdots \rightarrow (s_{o\omega}, s_{b\omega}) \rightarrow (s_{o\omega}, s_{b\omega})
\]
If a pair \( (s_a, s_b) \) is in an unresolvable chain, then it is unresolvable, and so are all its predecessors of all orders.

A set system \( \Theta \subseteq P \) is said to be “resolvable” if every \( (s_a, s_b) \) contained in any block of \( \Theta \) is resolvable.

Let us consider the sequential machine \( M_2 \) whose state transition table is shown in Fig. 3(a).

\( P = \{ (s_1, s_3), (s_2, s_4, s_5) \} \) is a compatible binary partition, and \( P' = \{ (s_1, s_2, s_6), (s_3, s_7) \} \) is a binary partition but is not compatible because \( f(s_3) = (s_3, s_7) \) which is contained in one block of \( P' \). For \( P \), let \( C(s_3) = 0 \) and \( C(s_4, s_5) = 1 \). Then, \( f_0(s_1, s_3) = (s_1, -) \) and \( f_1(s_1, s_3) = (s_2, s_3) \), and hence \( (s_1, s_3) \) is resolved. Similarly, \( (s_2, s_3) \) is also resolved for \( P \). Because \( f_0(s_2, s_4) = (s_3, s_7) \) and \( f_1(s_2, s_4) = (s_4, s_7) \), \( (s_2, s_4) \) is not resolved for \( P \); however, \( (s_2, s_4) \) is resolvable because both \( (s_1, s_3) \) and \( (s_2, s_3) \) are resolved. Similarly, \( (s_2, s_4) \) is not resolved but is resolvable.

As another example, let us consider the sequential machine \( M_3 \) whose state transition table is shown in Fig. 4. The binary partition \( P = \{ (s_1, s_4, s_5, s_6), (s_2, s_7) \} \) is compatible. Let \( C(s_3, s_8, s_9, s_7) = 0 \) and \( C(s_3, s_8, s_7) = 1 \). Then, \( f_0(s_5, s_5) = (s_4, s_6) \) and \( f_1(s_5, s_5) = (s_5, s_7) \), and hence \( (s_5, s_7) \) is resolved. Because \( f_0(s_5, s_7) = (s_5, s_7) \) and \( f_1(s_5, s_7) = (s_5, s_7) \) is 1-resolved but not resolvable. \( (s_5, s_7) \) is unresolvable because \( f_0(s_5, s_7) = (s_5, s_7) \) is unresolvable. \( (s_5, s_7) \) are all unresolvable because they form an unresolvable chain \( (s_5, s_7) \rightarrow (s_5, s_7) \rightarrow (s_5, s_7) \rightarrow (s_5, s_7) \).

Definition 9: Let \( P = \{B_0, B_1\} \) be a compatible binary partition and \( (s_a, s_b) \) contained in either \( B_0 \) or \( B_1 \). The “resolvability” of \( (s_a, s_b) \), denoted by \( R(s_a, s_b) \), is defined as follows: \( R(s_a, s_b) = 1 \) for every \( s_a \in S \); \( R(s_a, s_b) = 2 \) if \( s_a \neq s_b \) and \( (s_a, s_b) \) is resolved. If \( (s_a, s_b) \) is resolvable but not resolved, then
\[
R(s_a, s_b) = 1 + \max \{ R[f_0(s_a, s_b)], R[f_1(s_a, s_b)] \}
\]
(28)
If \( (s_a, s_b) \) is unresolvable, then \( R(s_a, s_b) = \infty \). The “resolvability” of a set system \( \Theta \subseteq P \), denoted by \( R(\Theta) \), is defined as
\[
R(\Theta) = \max \{ R(s_a, s_b) \mid (s_a, s_b) \}
\]
(29)
contained in a block of \( \Theta \).

It is noted that \( \Theta \) is “resolvable” if and only if \( R(\Theta) \) is finite. Furthermore, \( R(\Theta) = 1 \) if and only if \( \Theta \) is a null set system. For the compatible binary partition \( P = \{ (s_1, s_3), (s_2, s_4, s_5) \} \) of \( M_2 \), \( R(P) = 3 \). For the compatible binary partition \( P = \{ (s_1, s_2, s_4, s_5), (s_3, s_7) \} \) of \( M_3 \), \( R(P) = \infty \).

Theorem 4: Let \( P \) be a compatible binary partition and \( \Theta \subseteq P \) a SRUE set system. If \( R(\Theta) = k < \infty \), then for any integer \( p \), \( 0 \leq p < k \), and \( b) \ R[E^p(\Theta)] = 1 \) for all integers \( p \geq k \).

Theorem 5: The theorem is trivially true for \( p = 0 \) and \( p = k \). Assume that \( p = 1 \) and \( k > 1 \). Since \( R(\Theta) = k \), there exists a pair \( (s_{a'}, s_{b'}) \) contained in a block of \( \Theta \) such that \( R(s_{a'}, s_{b'}) = k \) and \( R(s_{a'}, s_{b'}) \leq k \) for every \( s_{a'}, s_{b'} \) contained in a block of \( \Theta \). Because \( \Theta \) is SRUE and \( R(s_{a'}, s_{b'}) = k \), both \( f_0(s_{a'}, s_{b'}) \) and \( f_1(s_{a'}, s_{b'}) \) must be in \( E(\Theta) \).

Furthermore, \( E^p(\Theta) \geq k + 1 \). On the other hand, assume that \( (s_{a'}, s_{b'}) \) is contained in a block of \( E(\Theta) \) and has the maximum resolvability among all pairs \( (s_{a'}, s_{b'}) \) in \( E(\Theta) \). Let \( k' = R(s_{a'}, s_{b'}) = R[E(\Theta)] \). Since both \( f_0^{-1}(s_{a'}, s_{b'}) \) and \( f_1^{-1}(s_{a'}, s_{b'}) \) are in \( \Theta \), \( \geq k + 1 \). Hence, \( k' = k - 1 \). This proves a) for the case \( p = 1 \). Suppose it is true for \( p = q < k - 1 \). Then, \( R[E^q(\Theta)] = k - q \). Furthermore, \( E^{q+1}(\Theta) \) is SRUE by virtue of Theorem 3, and certainly \( E^q(\Theta) \subseteq P \).

Because we have proved a) for \( p = 1 \), we obtain \( R[E^q(\Theta)] = R[E^{q+1}(\Theta)] = k - q - 1 \). Consequently, a) is true for \( p = q + 1 < k \), and this completes the proof for a). For b), we notice that \( E^{q+1}(\Theta) \) is a null set system; so are \( E^p(\Theta) \), \( p \geq k \). Hence, \( R[E^p(\Theta)] = 1 \) for all integers \( p \geq k - 1 \).
The following corollaries are obvious:

**Corollary 3:** If a set system \( \theta \) is \( k \)-SRUE and resolvable, then \( R[\gamma^{-1}(\theta)] \leq R(\theta) + p \) for any integer \( p \), \( 0 \leq p < k \).

**Corollary 4:** Let \( \theta \) be a null \( k \)-SRUE set system. Then, \( P = E^{-1+1}(\theta) \) is a compatible binary partition such that \( E^{-1}(P) = \theta \).

**Definition 10:** A sequential machine \( M \) is “unitarily realizable with a single shift register” (SPUR) of length \( m \) if there exists a null \( m \)-SRUE set system on the state set of \( M \).

**Theorem 5:** A sequential machine \( M \) is unitarily realizable with a single shift register of length \( m < \infty \) if and only if there exists a resolvable compatible binary partition \( P \) and \( R(P) \leq m \).

**Proof:** If \( M \) has a resolvable compatible binary partition \( P \) whose \( R(P) \leq m \), it follows from Theorem 3 and Corollary 2 that \( E^{-1}(P) \) is \( m \)-SRUE because \( P \) is always 1-SRUE. Since \( R(P) \leq m \), \( R(E^{-1}(P)) = 1 \). In other words, \( E^{-1}(P) \) is a null \( m \)-SRUE set system, and hence \( M \) is unitarily realizable with a single shift register of length \( m \). On the other hand, if \( M \) is unitarily realizable with a single shift register of length \( m \), it follows from Definition 10 that there exists a null \( m \)-SRUE set system \( \theta \). From Theorem 2 and Corollaries 1 and 4, we know that \( P = E^{-1+1}(\theta) \) is a compatible binary partition. Because \( R(\theta) = 1 \), it follows from Corollary 3 that \( R(P) \leq m \). Hence, the theorem.

In general, a sequential machine \( M \) may have none, one, or more than one compatible binary partition. If \( M \) has more than one compatible binary partition, their resolvabilities may be different. Let \( L \) be the set of all compatible binary partitions of \( M \) and \( R(M) = \min \{ R(P) \mid P \in L \} \). Then, the following corollary is obvious.

**Corollary 5:** If \( R(M) \) of a sequential machine \( M \) is \( m \), where \( m < \infty \), then \( M \) is unitarily realizable with any shift register of length at least \( m \), but not unitarily realizable with any shift register of length less than \( m \).

With the above results, we can now present an algorithm for unitary realization of a sequential machine with a single shift register of minimal length.

**Algorithm 1:** Given the state transition table of a sequential machine \( M \), find a unitary realization of \( M \) with a single shift register of minimal length.

1. Find the set \( L \) of all compatible binary partitions of \( M \) from its state transition table. If \( L = \{ \} \), \( i.e., \) \( L \) is empty, then \( M \) is not unitarily realizable with a single shift register. If \( L \neq \{ \} \), go to the next step.

2. Find the resolvabilities of all compatible binary partitions in \( L \). If the resolvability of every compatible binary partition in \( L \) is not finite, then \( M \) is not unitarily realizable with a single shift register. Otherwise, find a compatible binary partition \( P \) in \( L \) with the smallest resolvability \( m \), and go to the next step.

3. Assign an arbitrary coding \( C_1 \) to \( P \), and find the \((m - 1)\)-expansion \( E_{m-1}(P) \) of \( P \) with the coding \( C_1 \). Let

the resultant coding on \( E_{m-1}(P) \) be \( C_m \). Find \( m \) switching functions \( Y_j, j = 1, 2, \ldots, m \) for the coding \( C_m \). Then, \( M \) is unitarily realized with a single shift register of length \( m \). This terminates the algorithm.

It follows from Corollary 5 that the unitary realization of \( M \) with a single shift register obtained by the above algorithm is obviously of minimal length. Now, we would like to discuss the methods of finding the set \( L \) of all compatible binary partitions of \( M \), the resolvability of a compatible binary partition, and \((m - 1)\)-expansion of a resolvable compatible binary partition.

Two states \( s \) and \( s' \) of \( S \) are said to be “adjacent,” denoted by \( s \sim s' \), if there exists a state \( s'' \) in \( S \) such that \( s \in \{s'' \} \) and \( s' \in \{s'' \} \). In other words, \( s \sim s' \) if \( s \) and \( s' \) appear as two entries in one row of the state transition table. Two states \( s \) and \( s' \) are said to be “chain adjacent” if there exists a subset \( \{s_1, s_2, \ldots, s_p, s' \} \) of \( S \) such that \( s \sim s_1 \sim s_2 \sim \cdots \sim s_p \sim s' \). Let \( \Phi(s) \) be the set of all states of \( S \) chain adjacent to \( s \). Then, \( \Phi(s_1) = \Phi(s) \) if and only if \( s_1 \in \Phi(s) \). Hence, \( \Phi \) defines a partition \(^1 \) on \( S \).

**Definition 11:** A pair \( \Psi = \{B, B'\} \) of subsets of \( S \), not both empty, is “compatible” if for each \( s \in B \) all states adjacent to \( s \), excluding \( s \), are in \( B' \), and vice versa. \( \Psi = \{B, B'\} \) is “minimal compatible” if the removal of any element of \( B \) or \( B' \) yields a noncompatible pair.

From the above definition the following lemma is obvious.

**Lemma 2:** For any compatible pair \( \Psi = \{B, B'\} \) of subsets of \( S \), \( B \) and \( B' \) are disjoint.

Let the pair \( \{B, B'\} \) of subsets of \( S \) be minimal compatible, and \( s \in B \). For any \( s' \in \Phi(s) \), there exists a subset \( \{s_1, s_2, \ldots, s_p \} \subseteq \Phi(s) \) such that \( s = s_1 \sim s_2 \sim \cdots \sim s_p \sim s' \). Because \( s \in B \) and \( \{B, B'\} \) are compatible, \( \{s_1, j = 0, 2, 4, \ldots, p \text{ or } p + 1 \} \subseteq B \) and \( \{s_1, j = 1, 3, \ldots, p \text{ or } p + 1 \} \subseteq B' \). Consequently, \( B \cap B' \subseteq \Phi(s) \). Let \( A = \Phi(s) \cap B \) and \( A' = \Phi(s) \cap B' \). It is obvious that \( \{A, A'\} \) is compatible. However, since \( \{B, B'\} \) is minimal compatible, we know that \( A = B \) and \( A' = B' \). Consequently, we have the following lemmas.

**Lemma 3:** If a pair \( \{B, B'\} \) of subsets of \( S \) is minimal compatible and \( s \in B \cup B' \), then \( \Phi(s) = B \cup B' \).

**Lemma 4:** If \( \{A, A'\} \) and \( \{B, B'\} \) are two minimal compatible pairs of subsets of \( S \) and \( A \cap B \) is not empty, then \( A = B \) and \( A' = B' \).

**Definition 12:** Let \( \Psi_j = \{B_j, B'_j\}, j = 1, 2, \ldots, q \), be a set of minimal compatible pairs such that \( B_j \cap B'_j = B_j \cap B'_j \) for all \( j \neq k \). The “sum” of \( \Psi_1, \Psi_2, \ldots, \Psi_q \), denoted by \( \Psi_1 + \Psi_2 + \cdots + \Psi_q \), is the set of all possible pairs \( \{B, B'\} \) such that \( B = B_1 \cup B_2 \cup \cdots \cup B_q \), where \( B_j \) is either \( B_j \) or \( B'_j \), and \( B' \) is the union of the remaining \( q \) subsets.

**Theorem 6:** A binary partition \( P \) on \( S \) is compatible if and only if there exist \( q \) distinct minimal compatible pairs \( \Psi_j = \{B_j, B'_j\}, j = 1, 2, \ldots, q \), and \( \bigcup_{j=1}^{q}(B_j \cup B'_j) = S \) such that \( P \subseteq (\Psi_1 + \Psi_2 + \cdots + \Psi_q) \).

\(^1\) Nichols called the relation \( s \sim s' \) “row connected” and the partition defined by \( \Phi \) the “row partition” of \( S \).
Proof: Let $P = \{B, B'\}$. The "if" part of the theorem is obvious from Definitions 7, 11, and 12. To prove the "only if" part, assume $P$ is a compatible binary partition. It follows from Lemmas 3 and 4 that for any $s_1 \in S$, $B_1 = B' \cap \Phi(s_1)$ and $B'_1 = B' \cap \Phi(s_1)$ form a minimal compatible pair $\Psi_1 = \{B_1, B'_1\}$. If $S = \Phi(s_1)$, then the "only if" part is obviously true; otherwise, choose a state $s_0$, where $s_0 \notin \Phi(s_1)$. Then, $B_2 = B \cap \Phi(s_2)$ and $B'_2 = B' \cap \Phi(s_2)$ form another minimal compatible pair $\Psi_2 = \{B_2, B'_2\}$. If $\Phi(s_3) \cap \Phi(s_2) \neq S$, repeat this process. Suppose that after generating $\Psi_j, j = 1, 2, \ldots, q$, $\cup_{j=1}^q \Phi(s_j) = S$. It follows from Lemma 2 and Definition 12 that $P \in (\Psi_1 + \Psi_2 + \cdots + \Psi_q)$. This completes the proof of the theorem.

Because $B_j, B'_j, j = 1, 2, \ldots, q$, in the above theorem are all disjoint, we have the following corollary.

Corollary 6: If a compatible binary partition on $S$ is in the sum of $q$ minimal compatible pairs, then there are $2^q - 1$ different compatible binary partitions of $S$.

Now, we can formulate an algorithm for generating all compatible binary partitions on $S$.

Algorithm 2: Generate all compatible binary partitions on $S$ of a sequential machine $M$ from its state transition table.

1) Pick any state $s_1 \in S$ and find $\Phi(s_1)$ from the state transition table of $M$. Put $s_1$ in $B_1$; all the states adjacent to $s_1$, excluding $s_1$ itself, in $B'_1$; all the states adjacent to the states in $B'_1$, excluding those states in $B'_1$ themselves, in $B_1$, and so forth. If we end with $B_1$ and $B'_1$ that are not disjoint, then $M$ does not have any compatible binary partition. If $B_1$ and $B'_1$ are disjoint, then $\{B_1, B'_1\}$ is a minimal compatible pair and $B_1 \cup B'_1 = \Phi(s_1)$. Then, go to the next step.

2) Pick any state $s_2 \in S$ which is not included in any minimal compatible pair previously generated and repeat Step 1. If each $s_2 \in S$ is included in one of these minimal compatible pairs, go to the next step.

3) Let $\Psi = \{B_j, B'_j\}, j = 1, 2, \ldots, q$, be all the minimum compatible pairs obtained and, of course, $\Phi(s_3) \cup \Phi(s_2) \cup \cdots \cup \Phi(s_q) = S$. Then, find $\Psi_1 + \Psi_2 + \cdots + \Psi_q$, and the elements in $\Psi_1 + \Psi_2 + \cdots + \Psi_q$ are all the compatible binary partitions on $S$.

Since the binary partition $P$ on $S$ consisting of only two states is trivially compatible and always has $R(P) = 1$, we assume in the following discussion that $S$ has at least three states. In order to facilitate the presentation of the algorithm for finding the resolvability of a compatible binary partition, we state the following definition.

Definition 13: Let $P = \{B_0, B_1\}$ be a compatible binary partition on $S$. A pair of states $(s_i, s_j) \subseteq B_0, s_i \neq s_j, \delta = 0$ or 1, is called a "primary pair" of $P$. A primary pair $(s_i, s_j)$ of $P$ is a primary successor of $(s_i, s_j)$ of $P$ if $(s_i, s_j) = f(s_i, s_j)$ or $(s_i, s_j) = f(s_i, s_j)$. Let $\Gamma_j^{(1)}$ be the set of all primary successors of $\lambda_j$. That is,

$$\Gamma_j^{(1)} = \{s_j | s_j = f(s_i, s_j), \delta = 0, 1\}.$$  (30)

Find $\Gamma_j^{(1)}, j = 1, 2, \ldots, t$.

2) Define recursively that

$$\Gamma_j^{(t+1)} = \{s_j | s_j = f(s_i, s_j), \delta = 0, 1, \lambda_t \in \Gamma_j^{(t)}\}.$$  (31)

In case $\Gamma_j^{(t)} = \phi$, $\Gamma_j^{(t+1)}$ is defined to be $\phi$. Starting from $\Gamma_j^{(1)}$, $j = 1, 2, \ldots, t$, find all $\Gamma_j^{(t)}, t = 2, 3, \ldots$ until either there exists a $\lambda_j$ such that $\lambda_j \in \Gamma_j^{(t)}$, or $\Gamma_j^{(t+1)} = \phi, j = 1, 2, \ldots, t$. The former indicates that $P$ is not resolvable and hence $R(P) = \infty$; the latter shows $R(P) = \epsilon + 1$, where $\epsilon$ is the smallest $\epsilon$ satisfying the condition that $\Gamma_j^{(t)} = \phi, j = 1, 2, \ldots, t$.

The expansion $E(\theta)$ of a set system $\theta$ with a coding $C_s$ can easily be found from the definition of the expansion $E(T)$ of the transition graph $T$ of $\theta$ with the coding $C_s$. However, we would like to present an efficient algorithm for obtaining the $(m-1)$-expansion $E^{m-1}(P)$ of a compatible binary partition $P$ with the resolvability $m$, as required in Algorithm 1.

Algorithm 4: Find the $(m-1)$-expansion $E^{m-1}(P)$ of a compatible binary partition $P$ with the resolvability $m$.

Let $P = \{B_0^{(0)}, B_1^{(0)}\}$. Code $B_0^{(0)}$ with 0 and $B_1^{(0)}$ with 1. To find $E(P)$, we split $B_0^{(0)}$ to $B_1^{(1)} = \cap f(B_0^{(0)})$ and $B_1^{(1)} = \cap f(B_0^{(0)})$, and code $B_1^{(1)}$ and $B_1^{(0)}$ with 00 and 01, respectively. Similarly, $B_1^{(0)}$ is split to $B_1^{(2)} = \cap f(B_1^{(2)})$ and $B_1^{(2)} = \cap f(B_1^{(2)})$. Then, $B_1^{(1)}$ and $B_1^{(2)}$ are coded with 11 and 10, respectively. Then, $E(P) = \{B_1^{(1)}, B_1^{(2)}, B_1^{(3)}, B_1^{(4)}\}$ with the code words 00, 01, 11, and 10 for $B_1^{(3)}, B_1^{(4)}, B_1^{(5)}$, and $B_1^{(6)}$, respectively. In general, we obtain $E(P)$ from $E'(P), q = 0, 1, \cdots, m-2$ by using the following formulas.

$$B_j^{(q+1)} = B_j^{(q+1)} \cap f(B_j^{(q+1)})$$  (32a)

for $j = 1, 2, \ldots, q$, and

$$B_j^{(q+1)} = B_j^{(q+1)} \cap f(B_j^{(q+1)})$$  (32b)

for $j = 2^q + 1, 2^q + 2, \ldots, 2^{q+1}$, where $[j/2]$ is the smallest integer not smaller than $j/2$. The code word for $B_j^{(q+1)}$ is obtained by adding 0 as the last digit to the code word for $B_j^{(q+1)}$ when $j = 4(d + 1)$ and $1 + 4d$, where $d = 0, 1, 2, \ldots, 2^{q-1} - 1$. When $j = 2 + 4d$ and $3 + 4d$, where $d = 0, 1, 2, \ldots, 2^{q-1} - 1$, the code word for $B_j^{(q+1)}$ is obtained by adding 1 as the last digit to the code word for $B_j^{(q+1)}$. The above process can be tabulated in the form of Table I. The code word for $B_j^{(q+1)}$ can be obtained by tracing $B_j^{(q+1)}$ back to the block in $P$, from which it is derived. For instance, from Table I we obtain 1100 for the code word of $B_4^{(0)}$. It is noted that the code word for the blocks in $E^{q+1}(P)$ always form a complete Gray code of $q + 2$ digits with respect to the subscripts of the blocks of $E^{q+1}(P)$. Furthermore, it...
TABLE I
GENERATION OF $E^{m-1}(P)$ OF A COMPATIBLE BINARY PARTITION $P$ WITH RESOLVABILITY $m$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$E(P)$</th>
<th>$E^1(P)$</th>
<th>$E^2(P)$</th>
<th>$E^3(P)$</th>
<th>$E^4(P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: $B_1^{(0)}$</td>
<td>$0: B_1^{(0)} = B_1^{(0)} \cap f(B_1^{(0)})$</td>
<td>$0: B_1^{(0)} = B_1^{(0)} \cap f(B_1^{(0)})$</td>
<td>$1: B_2^{(0)} = B_2^{(0)} \cap f(B_2^{(0)})$</td>
<td>$1: B_1^{(0)} = B_1^{(0)} \cap f(B_1^{(0)})$</td>
<td>$0: B_2^{(0)} = B_2^{(0)} \cap f(B_2^{(0)})$</td>
</tr>
<tr>
<td>1: $B_2^{(0)}$</td>
<td>$1: B_2^{(0)} = B_2^{(0)} \cap f(B_2^{(0)})$</td>
<td>$0: B_1^{(0)} = B_1^{(0)} \cap f(B_1^{(0)})$</td>
<td>$1: B_1^{(0)} = B_1^{(0)} \cap f(B_1^{(0)})$</td>
<td>$1: B_2^{(0)} = B_2^{(0)} \cap f(B_2^{(0)})$</td>
<td>$0: B_2^{(0)} = B_2^{(0)} \cap f(B_2^{(0)})$</td>
</tr>
</tbody>
</table>

Fig. 5. (a) The state transition table of machine $M_4$. (b) The transition graph of $E^2(P')$ with the coding $C_3$ for $M_4$ where $P' = \{(s_1, s_2, s_3)\}$ and $R(P') = 3$, and $C_3$ is obtained from Table III. (c) The coded state transition table of $M_4$ according to $C_3$ and the transition graph of $E^2(P')$. (d) A unitary single shift-register realization of $M_4$. 

---

$S_{\alpha} = \{(s_1, s_2, s_3)\}$, $s_1 = 00$, $s_2 = 01$, $s_3 = 11$, $s_4 = 10$

---

$Y_1 = \bar{Y_2} \cdot Y_{3}$

---

$\text{DELAY}$ $\text{NOT Gate}$ $\text{AND Gate}$ $\text{OR Gate}$
follows from Theorem 4 that $E^{(m-1)}(P)$ is a null $m$-SRUE set system.

The above algorithms are illustrated by the following example. Consider the sequential machine $M_4$ whose state transition table is shown in Fig. 5. Following Algorithm 2, we obtain that $\Psi_1 = \{(s_1), (s_3, s_4)\}$ and $\Psi_2 = \{(s_2), (s_5, s_6)\}$. There are only two compatible binary partitions $P = \{(s_1, s_2), (s_3, s_4, s_5, s_6)\}$ and $P' = \{(s_1, s_2, s_3), (s_4, s_5, s_6)\}$ in $\Psi_1 + \Psi_2$. Following Algorithm 3, we find that $P$ is not resolvable because the primary pair $(s_1, s_2)$ is a primary successor of itself. However, we find that $R(P') = 3$ because $\Gamma_{P'}^{(2)} = \phi$ for each $\lambda_j$. All $\lambda_j, \Gamma_{P'}^{(1)}$, and $\Gamma_{P'}^{(2)}$ for $P'$ are listed in Table II. Following Algorithm 4, we obtain $E(P') = \{(s_1), (s_3), (s_4, s_5), (s_2, s_6)\}$ as shown in Table III. For the sake of convenience, we have listed $C_3$ as the last column of Table III. Following the coding $C_3$ and the state transition table of $M_4$, we obtain the transition graph of $E^3(P')$ shown in Fig. 5(b), which results in the coded state transition table of $M_4$ shown in Fig. 5(c). In Fig. 5(c), the code words with asterisks are filled for the unspecified entries in order to simplify the switching functions $Y_j, j = 1, 2, 3$. These switching functions are

$$Y_1 = x_1x_2y_2 + x_1x_2y_2 + x_1y_2y_3 + x_1y_2y_3$$

$$Y_j = y_{j-1}, \quad j = 2, 3. \quad (33)$$

The resultant realization of $M_4$ is shown in Fig. 5(d).

It is noted that the coded state transition table of a sequential machine can be found directly from the unitary coding obtained from the $(m-1)$-expansion of a compatible binary partition with resolvability $m$, since the columns corresponding to each $Y_j, j = 2, 3, \ldots, m$, under all inputs must be identical in order to have a single shift-register realization.

**IV. Multiple Shift-Register Realizations of Sequential Machines**

If a sequential machine $M$ is not realizable with a single shift register, then it is natural to consider its realization with more than one shift register. Since an $r$-state sequential machine $M$ is always realizable with $m$ memory elements, where $m$ is the smallest integer not smaller than $\log_2 r$, a trivial $m$-shift-register realization always exists. In this section, we shall consider the unitary realization of a sequential machine with a minimal number of shift registers. The sequential machines considered in this section are assumed to be not SRUR, unless specified otherwise.

Consider a sequential machine $M$ with a coding $C$ such that the memory elements are arranged as $l$ shift registers $SR(j)$ of length $m_j, j = 1, 2, \ldots, l$. For each $SR(j)$, the $m_j$ state variables $y_{j,1}, y_{j,2}, \ldots, y_{j,m_j}$ in $SR(j)$ have $2^{m_j}$ different $m_j$-tuples. If the states of $M$ with the same $m_j$-tuple in their code words are grouped together, then the set of all such groups (some groups may be empty) is a set system $\theta_j'$ on the state set $S$ of $M$. If $\theta_j$ is a set system obtained from $\theta_j'$ by deleting all duplicating groups, then $\theta_j$ is $m_j$-SRUR. Because each code word of $C$ is assigned to at most one state of $M$, the product $\theta_1\theta_2\cdots\theta_l$ is a null set system on $S$, where $\theta_1\theta_2\cdots\theta_l$ is a collection of blocks obtained by taking the intersection of all possible combinations of the blocks, one from each $\theta_j, j = 1, 2, \ldots, l$.

From the above observations, one of the approaches to the problem of multiple shift-register realizations of sequential machines is to find a set of SRUE set systems $\theta_j, j = 1, 2, \ldots, l$, such that their product is a null set system. However, we shall define unitary realization for the multiple shift-register realization case in a more restrictive way.

Let $P = \{B_1, B_2, \ldots, B_n\}$ be a partition on the state set $S$ of a sequential machine $M$. The "skeleton matrix" $K(P) = [k_{i,j}]$ is a matrix of order $n$ whose entry $k_{i,j}$ is 1 if there exist two states $s_{E} \in B_i$ and $s_{F} \in B_j$ and an input $i_j$ of $M$ such that $s_{E} = f(s_{E}, i_j)$, and is 0 otherwise. Let $G[K(P)]$ be the set of sequential machines with the skeleton matrix $K(P)$. A partition $P$ on $S$ of $M$ is "unitarily realizable with a single shift register" ("SRUR") if any sequential machine in $G[K(P)]$ is SRUR.

**Definition 14**: A sequential machine $M$ is "unitarily realizable with $l$ shift registers" ("$l$-SRUR") if there...
exist \( l \) SRUR partitions \( P_j, j = 1, 2, \ldots, l \), on the state set \( S \) of \( M \) such that the product \( P_1 \cdot P_2 \cdot \cdots \cdot P_l \) is a null partition.

It is obvious that a partition \( P \) is not SRUR if \( K(P) \) has more than two 1's in any row. In order to find all SRUR partitions on \( S \) of \( M \), we first sequentially form all the partitions on \( S \) each of which has no more than two 1's in a row of its skeleton matrix. Then, each of these partitions is tested for unitary realizability with a single shift register by using the algorithms presented in the previous section. It is noted that because every binary partition on \( S \) of \( M \) is SRUR, we only need to test those partitions on \( S \), each of which has at least three blocks and no more than two 1's in a row of its skeleton matrix. Hence, we shall formulate an algorithm for obtaining the set of all SRUR partitions on \( S \) of \( M \) with at least three blocks. After all SRUR partitions, including those with only two blocks, are obtained, choose a minimum number of them such that their product is a null partition.

Let \( P_0 = \{ B_1, B_2, \ldots, B_n \}, n \geq 3 \) be a partition on \( S \). For any \((B_1) \in P_0\), let \( f[B_1] = \{ B_2, B_3 \in P_0 \) and contain at least one state of \( f(B_1) \). Then, form a class \( F(P_0) = \{ \mu_1, \mu_2, \ldots, \mu_n \} \) from \( f[B_1], j = 1, 2, \ldots, n, \) as follows.

For each \( f[B_1] \) consisting of at least three blocks of \( P_0 \) there exists a \( \mu_i \in F(P_0) \) such that \( \mu_i \supseteq f[B_1] \), and for each \( \mu_i \), there exists an \( f[B_1] \) consisting of at least three blocks of \( P_0 \) such that \( \mu_i = f[B_1] \). It is noted that \( F(P_0) = \phi \) if and only if each \( f[B_1] \) consists of at most two blocks of \( P_0 \).

First, let us consider the case when \( F(P_0) \neq \phi \). Let \( h_i \) be the number of blocks in \( \mu_i, i = 1, 2, \ldots, \lambda \). The \( h_i \) blocks in \( \mu_i \) can be grouped in a pair of nonempty disjoint subsets \( B_{a1}, B_{a2} \), and there are \( d_i \) such pairs, where \( d_i = 2^{n-1} - 1 \). For a set \( \{ (b_{a1}, b_{a2}), (b_{b1}, b_{b2}), \ldots, (b_{al}, b_{l}) \} \), we construct a partition \( P \) using the following rules.

1. If states \( s' \) and \( s'' \) are in one block of \( P_0 \), then they are placed in one block of \( P \). 2. If blocks \( B_j \) and \( B_k \) of \( P_0 \) are contained in any subset \( b_{a1}, b_{a2}, \ldots, b_{al}, b_{l} \) \( \lambda \) then all states in \( B_j \) and \( B_k \) are placed in one block of \( P \). 3. No two states are placed in the same block of \( P \) unless they are placed in a block under rule 1 or 2). The partitions obtained from \( P_0 \) and all possible sets \( \{ (b_{a1}, b_{a2}), (b_{b1}, b_{b2}), \ldots, (b_{al}, b_{l}) \} \) are called “partitions of Type I mod (\( P_0 \))”, and the set of all partitions of Type I mod (\( P_0 \)) is denoted by \( U(P_0) \). It is noted that \( P_0 \in U(P_0) \) when \( F(P_0) \neq \phi \), and \( P_0 \) is the only partition in \( U(P_0) \) when \( F(P_0) = \phi \).

Theorem 7: Let \( P_0 = \{ B_1, B_2, \ldots, B_n \} \) and \( P' = \{ B'_1, B'_2, \ldots, B'_n \} \) be two partitions on the state set \( S \) of a sequential machine \( M \). If \( F(P_0) \neq \phi \), \( F(P') = \phi \), and \( P' \supseteq P_0 \), then there exists a partition \( P \in U(P_0) \) such that \( P' \supseteq P \).

Proof: Because \( F(P_0) \neq \phi \), there exists a block \( B_j \in P_0 \) such that \( f[B_j] \) consists of at least three blocks of \( P_0 \). By relabeling the blocks of \( P_0 \), we can assume that \( B_j \in F(P_0) \) for \( j = 1, 2, \ldots, \lambda \) and for \( j = \lambda + 1, \lambda + 2, \ldots, n \) either \( f[B_j] \) has less than three blocks or \( f[B_j] 

Next, let us consider the case when \( F(P_0) = \phi \) or \( U(P_0) = P_0 \). By merging any two blocks of \( P_0 \) into one block we obtain a partition with \( n - 1 \) blocks. There are \( n(n - 1)/2 \) such partitions, called “partitions of Type II mod \( P_0 \).” Let \( U_0(P_0) \) be the set of all partitions of Type II mod \( P_0 \). It is obvious that \( P_0 \in U_0(P_0) \), and that if a partition \( P' \supseteq P_0 \), then there exists a partition \( P \in U_0(P_0) \) such that \( P' \supseteq P \). Let \( U(P) = U(P_0) \) when \( F(P) = \phi \), and \( U(P) = U(P_0) \) otherwise. Let \( P' \) be a partition on \( S \) of \( M \) for which \( F(P') = \phi \). It follows from Theorem 7 and the definition of \( U(P_0) \) that if \( P' \supseteq P_0 \), then there exists a partition \( P \in U(P_0) \) such that \( P' \supseteq P \). If \( P' \neq P \), then \( P' \supseteq P \) and we can use the same argument to show that there exists a partition \( P'' \in U(P) \) such that \( P'' \supseteq P'' \). Hence, we have the following theorem.

Theorem 8: Let \( P_0 = \{ B_1, B_2, \ldots, B_n \} \) be a partition on the state set \( S \) of a sequential machine \( M \), and let \( U(P_0) = U_0(P_0) \) when \( F(P_0) = \phi \), and \( U(P) = U(P_0) \) otherwise. Let \( P' \) be a partition on \( S \) of \( M \) for which \( F(P') = \phi \). If \( P' \supseteq P_0 \), then there exists a partition \( P' \in U(P_0) \) such that \( P'' \in U(P) \) and \( P'' \supseteq P'' \).

Based on the above theorem, we can formulate the following algorithm.

Algorithm 5: Given a sequential machine \( M \) which is not SRUR, find the set \( J \) of all SRUR partitions of \( M \), each with at least three blocks.

1. Let \( P_0 \) be the null partition on the state set \( S \). Find all the partitions in \( U(P_0) \) or find all the partitions in \( U_0(P_0) \) when \( U_0(P_0) = P_0 \). Put all those partitions each with at least three blocks in a set \( U \). Set \( J = \phi \).

2. If \( U = \phi \), go to step 4). If \( U \neq \phi \), choose any \( P \in U \) and delete \( P \) from \( U \). If \( F(P) \neq \phi \), add all partitions of Type I mod \( P \) each with at least three blocks to \( U \), and repeat this step. Otherwise, go to step 3).

3. Add all partitions of Type II mod \( P \) each with
at least three blocks to $U$, and then test $P$ for SRUR. If $P$ is not SRUR, go to step 2). If $P$ is SRUR, add $P$ to $J$ and go to step 2).

4) $J$ is the set of all SRUR partitions of $M$ each with at least three blocks.

All SRUR partitions of a sequential machine $M$ each with at least three blocks can be obtained by Algorithm 5. Since all binary partitions are trivially SRUR, any of them may be used in realizing $M$. In order to realize $M$ with a minimum number of shift registers, we have to choose a set consisting of a minimum number of SRUR partitions such that the product of the SRUR partitions of the set is null. Unfortunately, there is no general selection procedure which is not exhaustive. However, imposing additional constraints may appreciably improve the situation. Nichols, for example, assumed that the maximal number $g_{\text{max}}$ of memory elements allowed in any realization of $M$ is specified, and then started finding those realizations, each consisting of the minimum number of shift registers. He presented an algorithm which successively eliminates undesirable solutions. With “proper” value of $g_{\text{max}}$, his algorithm may effectively find “fairly good” solutions. For this part, the reader is referred to Nichols. [3] Designers with other design constraints or criteria may formulate other selection procedures.

To illustrate the unitary realization method with a minimum number of shift registers, let us consider the sequential machine $M_9$ whose state transition table is shown in Fig. 6(a). Following Algorithm 5, we find that $M_9$ has 20 SRUR partitions, each of which has at least three blocks. They are listed in Table IV. If we specify $g_{\text{max}} \geq 5$, then all the 20 SRUR partitions are acceptable for realizing $M_9$. However, if we specify $g_{\text{max}} = 4$, then only $P_1$, $P_2$, $P_4$, $P_9$, and $P_{13}$ are acceptable because $P_8$, for example, requires at least 2 state variables to code its blocks and at least 3 state variables to code the states in its largest block. Since there are only 5 partitions acceptable when we specify $g_{\text{max}} = 4$, it is easy to select a minimum number of these partitions such that their product is null. In this case, we find that $P_1 \cdot P_{13}$ is a null product involving the minimum number of SRUR partitions. Then, from the skeleton matrices of $P_1$ and $P_{13}$ and the method presented in the last section, we obtain the following codings for $P_1$ and $P_{13}$:

$P_1 : (s_1, s_6, s_9), (s_2, s_6, s_7), (s_3, s_4, s_9)$

$y_{12} \rightarrow 00, 01, 10, 11$

$P_{13} : (s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)$

$y_{34} \rightarrow 00, 10, 11, 01.$

From these codings, we have the state assignment for $M_9$ shown in Fig. 6(b), which yields the following switching functions:

$Y_1 = x_1 y_1 y_2 (x_2 + y_2) + x_1 (y_1 + y_2) + y_1 (x_2 + y_2)]$

$y_2 = x_2 y_2 (x_1 + y_2) + x_2 (x_1 y_2 + y_2),$

$Y_3 = x_3 (x_1 y_2 + y_2), y_3 y_2 (x_2 + x_1 y_2),$

$Y_4 = y_4 (x_1 y_2 + x_2 y_2 + y_2 + y_2) - j, j = 2, 4.$

(a)

<table>
<thead>
<tr>
<th>State</th>
<th>Assigned Code Words (y_0y_3y_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0000, 1010</td>
</tr>
<tr>
<td>01</td>
<td>1010, 1011</td>
</tr>
<tr>
<td>10</td>
<td>1101</td>
</tr>
<tr>
<td>11</td>
<td>1111, 1110</td>
</tr>
<tr>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>13</td>
<td>0011, 0100, 0110, 0111</td>
</tr>
<tr>
<td>17</td>
<td>1000</td>
</tr>
<tr>
<td>18</td>
<td>1100</td>
</tr>
<tr>
<td>19</td>
<td>0001, 0101</td>
</tr>
</tbody>
</table>

(b)

Fig. 6. (a) The state transition table of machine $M_9$. (b) A unitary state assignment for $M_9$ obtained from $P_1$ and $P_{13}$.

**Table IV**

All SRUR Partitions of $M_9$, Each with at Least Three Blocks

$P_1 = \{(s_1, s_4, s_9), (s_6, s_5, s_7), (s_8, s_5, s_9)\}$

$P_2 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_3 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_5, s_9)\}$

$P_4 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_5 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_6 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_7 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_8 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_9 = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_{10} = \{(s_1, s_4, s_9), (s_5, s_6, s_7), (s_8, s_6, s_9)\}$

$P_11 = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{12} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{13} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{14} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{15} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{16} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{17} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{18} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{19} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$

$P_{20} = \{(s_1, s_7, s_9), (s_2, s_4, s_6), (s_3, s_5, s_9)\}$
The realization of this set of switching functions obviously yields two shift registers, each of length 2. Furthermore, it is a realization of $M_6$ with a minimum number of shift registers and state variables because $M_6$ is not single shift-register realizable\(^1\) and has 9 states.

V. CONCLUSIONS AND FURTHER PROBLEMS

In this paper, a many-to-one state assignment method for shift-register realizations of sequential machines has been presented. For single shift-register realization, this method is based on finding a compatible binary partition $P$ with the smallest resolvability $m$, and the coding, which yields a realization with a single shift register of minimal length, is obtained directly from the $(m-1)$-expansion of $P$. It is noted that the one-to-one state assignment method for shift-register realization proposed by Nichols\(^1\) can be considered as a special case of ours, since his method of generating a shift-register partition $P_s$ of minimal length from another shift-register partition of $P_s$ is essentially to take the $1$-expansion of $P_s$. However, his procedure does not allow expansion of a set system and cannot be applied to take the expansion operation repeatedly as we proposed in Algorithm 4. Further, in order to find a minimal shift-register realization in Nichols\(^1\) all shift-register partitions of lengths 2, 3, ..., must be generated in that order. With Algorithm 5 presented in this paper each SRUR partition is generated practically independent of all other SRUR partitions, and SRUR partitions with a large number of blocks are usually generated first. This property is desirable because a minimal shift-register realization often consists of a small number of SRUR partitions with many blocks. The effort of generating SRUR partitions with few blocks can be saved when the process of generating SRUR partitions and the process of constructing minimal shift-register realization are performed concurrently, because a satisfactory shift-register realization is often found before all SRUR partitions are generated. All the algorithms presented in this paper can easily be programmed on digital computers, and efficient programming for all these algorithms has been written for CDC 3400. However, except for Algorithm 5, all the algorithms are suitable for hand computation.

It is obvious that there exist sequential machines which can be realized in single shift-register forms which are not unitary. For instance, the sequential machine $M_6$, whose state transition table is shown in Fig. 7, has no compatible binary partitions because $\Phi(s_1) = \{s_1, s_2, s_3, s_7\}$ cannot be grouped into a minimal compatible pair of subsets (Theorem 6). Hence, it is not realizable in a single shift-register form with any unitary coding. However, if we split state $s_1$ into two equivalent states $s'_1$ and $s''_1$, and define $f(s'_1, i_2) = f(s''_1, i_2) = f(s_1, i_2)$, $j = 1, 2, f(s_9, i_2) = s'_1$ and $f(s_10, i_2) = s''_1$, then the binary partition \{($s'_1$, $s_2$, $s_3$, $s_7$), ($s''_1$, $s_2$, $s_3$, $s_7$, $s_9$, $s_10$)\} is compatible and has resolvability of 8. Hence there exists a unitary coding $C$ on \{$s'_1$, $s''_1$, $s_2$, $s_3$, $s_7$, $s_9$, $s_10$\} such that $M_6$ can be realized with a single shift register. In general, if a sequential machine with a coding $C$ is realizable with a single shift register, then we can always expand the state transition table of the sequential machine such that a resolvable compatible binary partition of the expanded state transition table exists. Hence, a possible approach to the general many-to-one state assignment method for single shift-register realization of sequential machines is to search for resolvable binary set systems of sequential machines. Similarly, for the multiple shift-register realization we may extend our method by looking for shift-register set systems instead of SRUR partitions.

REFERENCES