SPECIFICATION AND REALISATION OF LOGIC CONTROL PROCEDURES ON THE BASIS OF PRESCRIBED INPUT-OUTPUT CHANGES

P. Arató

Department of Process Control,
Technical University, H-1521 Budapest

Received December 10, 1986

Abstract

In this report, a method is outlined for handling the logic control procedures both in the specification level and on the hardware implementation level of the design. The heuristic and intuitive character of constructing the flow chart and defining the states has been reduced to a large extent.

This method may result in several kinds of uniform hardware structures for either synchronous or asynchronous control units initially specified only by input-output sequences. Introducing differential mappings for the description of sequential operation, the prescribed sequences for input and output changes can be considered as the initial specification of a control unit. This specification yields a so-called $B:K$ table and a $B:K$ graph as representation of the required operation.

The definition of the states is made by interpreting the compatibility relation between the prescribed output changes.

The procedure of the state definition results in the $B:K:A$ set or graph which corresponds to the minimised flow table obtained from the state reduction of incompletely specified sequential circuits. The properties of the canonical $B:K$ set and graph always ensure the existence of an optimal cover. If the fixed hardware structure contains flip-flops for storing the output combination, then the influence of these flip-flops on the state reduction are automatically taken into consideration by the method outlined in the report.

Also, by the introduction of an optimal cover for the identifying functions related to the output changes, the logical expressions for the realisation of the hardware can be simplified.

The specification and description method, outlined in this report, has the advantage of defining the prescribed sequences of input and output changes in separate fragments. Applying the prescribed input section changes, these separate fragments can be joined together and the $B:K$ set can be calculated systematically. In this way, the specification for the synthesis procedure may become more rigorous than it was initially. However, it is not necessary to form a coherent specification by intuition.

CONTENTS

Introduction

A logic control procedure is realised by a sequential circuit which can be considered as an abstract automaton defined by the mappings f_z and f_y as follows: [1] [4]

$$f_z(X, y) \Rightarrow Z$$

$$f_y(X, y) \Rightarrow Y$$

where X is the set of input combinations,
Z is the set of output combinations,
y is the set of secondary combinations actually present at the input and so representing the present state of the automaton,
Y is the set of secondary combinations actually present at the output and so representing the next state of the automaton.

There is a correspondence indicating which elements of the sets X and y in the arguments are mapped onto which elements of the sets Z and Y, respectively. Only this correspondence specifies the automaton or sequential circuit.

The model $f_z(X, y) \Rightarrow Z$ is called the Mealy-type [1] and it can be transformed into the Moore-type model [1], in which the output combinations (Z) are not directly influenced by the input combinations as formalized by a mapping $f'_z(y) \Rightarrow Z$.

In Figure 1, the dashed units in the feedback lines are D-type flip-flops. In the case of synchronous sequential circuits, they periodically open the feedback lines.

The feedback lines of an asynchronous circuit are always closed.

The flow table [1] or a graph-representation [1] of the mappings f_z and f_y does not uniquely characterize a sequential circuit unless it is known whether the circuit is synchronous or asynchronous.

The classical analysis and synthesis procedures, based on the flow table or graph, are hard to handle—even by computer—in the case of sequential circuits with a large number of inputs. Since the number of input combinations increases exponentially with the number of input variables, this problem arises even in the case of more than five or six inputs. Another difficulty, caused by a

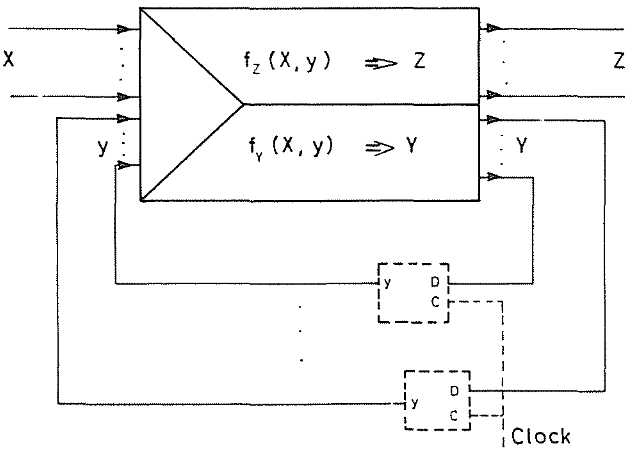


Fig. 1

large number of inputs, is the handling of the "don't care" combinations, the number of which also grows very quickly with the number of inputs.

In practice, the control units of digital systems can be characterized—even in simple cases—by a relatively large number of inputs and outputs. One reason for this is that the control unit must communicate with the environment as well as with the functional units of the system. This communication, described by the *control procedure*, generally requires—even in simple systems—many inputs and outputs from the control unit.

For this reason, the practical methods for the synthesis of control units or sequential circuits, whith a large number of inputs, are based mainly on various kinds of*flow-chart* [2] [3] [6] [7] [8]. These flow-charts can be considered as a directed graph representation of the control procedure. The construction of the flow-charts and the definition of the states on them are basically heuristic and intuitive.

The flow chart is the initial description of the control procedure for the most usual hardware realizations having fixed uniform structure [2] [3] [5] [6] [7] [8].

One of these uniform fixed structures is shown in Fig. 2 and is called the synchronous phase register structure [6] [8]. The combinational part (C) may be realised by memory units. The phase register and the synchronizing flip-flops and the clock-enabling network can be considered to be drawn together as a processor. This yields a kind of microprogrammed structure according to Fig. 3.

Further on, a method will be outlined, which eliminates to a large extent the heuristic and intuitive feature of the construction of the flow chart and the definition of the states (phases). This method may result, in a mostly systematic

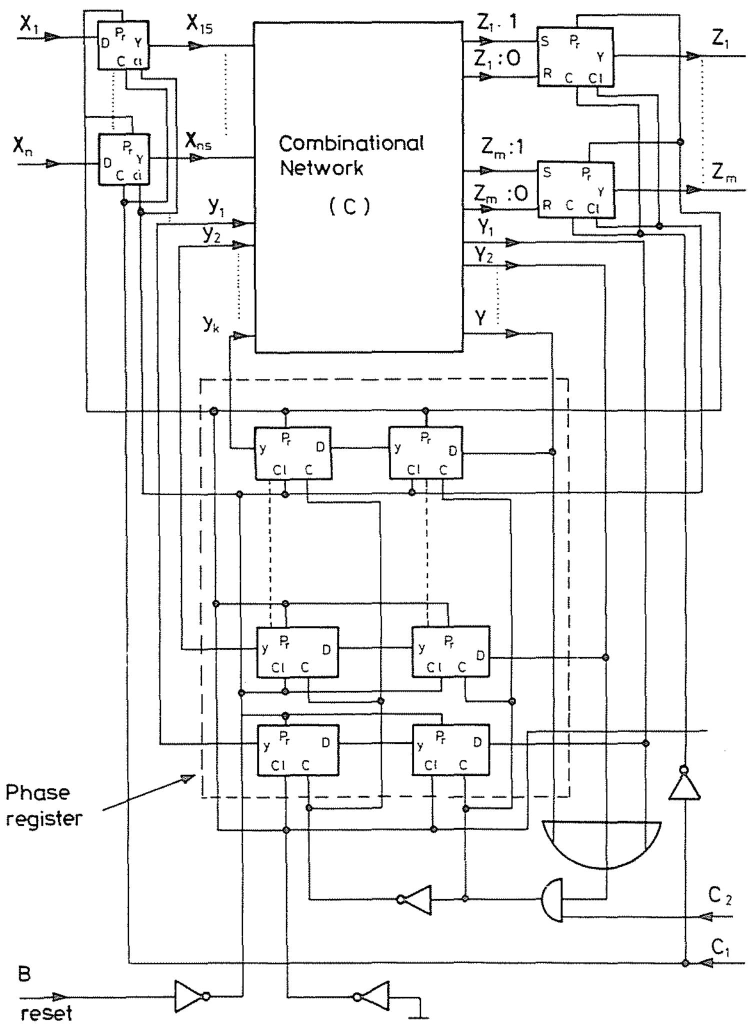


Fig. 2

way, in several kinds of uniform phase register structures for either synchronous or asynchronous control units initially specified only by input-output sequences.

For this purpose, the mappings f_y and f_z will be represented by mappings of the changes between the input, output and secondary combinations.

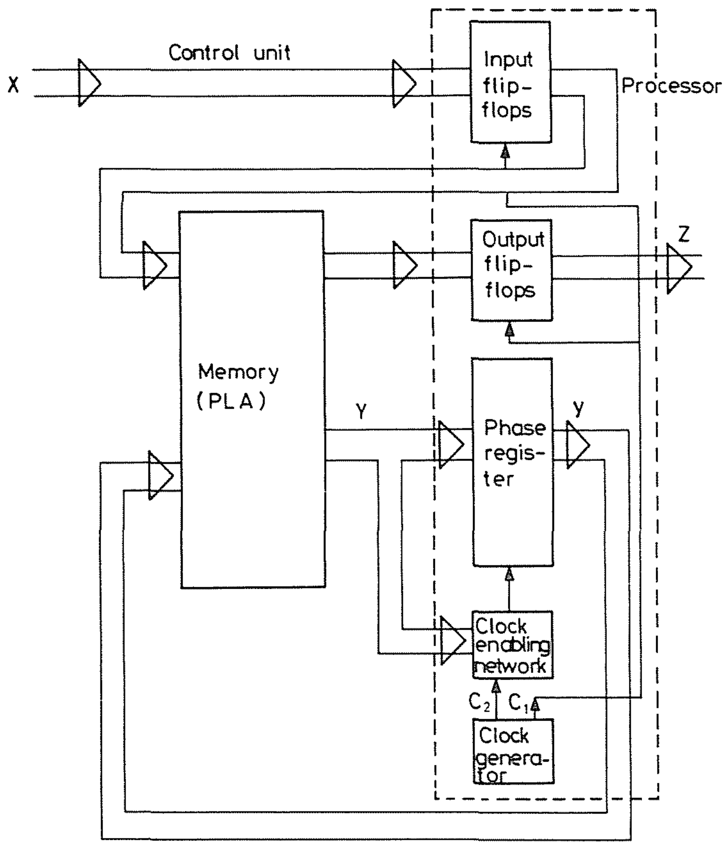


Fig. 3

I. The differential mapping

In the specification of a control procedure, the sequences of output and secondary combinations are generally prescribed as the response to only a few of the possible sequences of input combinations. In this sense, a given sequence of input, output or secondary combinations can be said to be *prescribed* or not prescribed. The sequences of input combinations, not prescribed by the control task, may cause arbitrary sequences of output and secondary combinations and can be considered as "don't care" sequences. Similarly, the sequences of output and secondary combinations, which are not prescribed for any of the sequences of input combinations, can also be handled as "don't care" sequences.

Let a single combination (input, output or secondary) be defined as a *prescribed* one, if it occurs in at least one of the prescribed sequences.

Let the differential mappings φ_z and φ_y be introduced as follows:

$$\varphi_z(B, y) \Rightarrow K \qquad \varphi_y(B, y) \Rightarrow A\,,$$

where

$\left\{\begin{matrix} B \\ K \\ A \end{matrix}\right\}$ denotes the set of prescribed combination-pairs, the elements of which occur adjacently in at least one of the prescribed sequences of

$$\left\{\begin{matrix} \text{input} \\ \text{output} \\ \text{secondary} \end{matrix}\right\} \text{combinations.}$$

Let the set $\left\{\begin{matrix} B \\ K \\ A \end{matrix}\right\}$ be called the set of the

$$\textit{prescribed} \left\{\begin{matrix} \text{input} \\ \text{output} \\ \text{secondary} \end{matrix}\right\} \text{changes.}$$

Where the symbols $\{\quad\}$ are used, it is intended to avoid repetition. Thus, in

$$\left\{\begin{matrix} B \\ K \\ A \end{matrix}\right\} \text{denotes} \left\{\begin{matrix} \text{input} \\ \text{output} \\ \text{secondary} \end{matrix}\right\}$$

B is intended to be read with input; K with output and A with secondary.

A control unit should produce a prescribed response only for prescribed input changes. In other words, it can be assumed that the control unit receives only prescribed input changes starting with an input combination related to the initial state.

Let $\left\{\begin{matrix} B_i^j \\ K_n^m \end{matrix}\right\}$ denote the prescribed $\left\{\begin{matrix} \text{input} \\ \text{output} \end{matrix}\right\}$ change which is realised if the $\left\{\begin{matrix} \text{input} \\ \text{output} \end{matrix}\right\}$ combination changes $\left\{\begin{matrix} \text{from } X^i & \text{to } X^j \\ \text{from } Z^n & \text{to } Z^m \end{matrix}\right\}$

Let A_s^q denote the prescribed secondary change, during which the control unit executes a state transition from the present state characterised by the secondary combination y^s to the next state characterised by the secondary combination Y^q.

The meaning of the differential mappings φ_z and φ_y can be illustrated as follows:

If $\varphi_z(B_i^j, y^k) \Rightarrow K_p^r$ and $\varphi_y(B_i^j, y^k) \Rightarrow A_k^t$ are given, then

$$f_z(X^i, y^k) \Rightarrow Z^p \qquad f_y(X^i, y^s) \Rightarrow Y^k$$

$$f_z(X^j, y^k) \Rightarrow Z^r \qquad f_y(X^j, y^k) \Rightarrow Y^t$$

are assumed.

This means that y^k in argument of φ_z and φ_y denotes always the present-state secondary combination at the beginning of B_i^j. This y^k is assumed to be produced as next state secondary combination by the predecessor prescribed input change at the occurrence of X^i starting from the predecessor present state y^s.

The differential mappings illustrated above do not distinguish the synchronously working control units from the asynchronously working ones, because φ_y does not make any restriction on the result of the mapping f_y when, for example, X^i occurs during y^k. For synchronous control units, the next-state secondary combination produced by $f_y(X^i, y^k)$ can be considered as a "don't care" one [1]. In asynchronous cases however, the normal fundamental mode [4] requires $f_y(X^i, y^k) \Rightarrow Y^k$.

Though the principles of Bochmann and Posthoff [11] for the differential vectors in the Boolean space are mathematically very sound, their approach has not been used in introducing the differential mapping and handling the changes of combinations in this paper. For, the method outlined in this paper does not interpret the difference $\mathrm{d}X^{i,j} = X^i \oplus X^j$, because the directions of changes of each signal and the starting and ending combinations are important in several steps of the method. So, the difference $\mathrm{d}X^{i,j}$ alone could not anyway represent the change from X^i to X^j.

For example, let the prescribed sequences of input, output and secondary combinations be given and let us assume a correspondence between them as follows:

$$X^i \; X^j \; X^n \; X^q \; X^h$$

$$Z^a \; Z^b \; Z^c \; Z^d \; Z^e$$

$$Y^k \; Y^t \; Y^p \; Y^u \; Y^r$$

Using the notation for the changes of combinations:

$$B_i^j \; B_j^n \; B_n^q \; B_q^h$$

$$K_a^b \; K_b^c \; K_c^d \; K_d^e$$

$$A_k^t \; A_t^p \; A_p^u \; A_u^r$$

If y^s is assumed to be the initial state, then the prescribed sequences of combinations given above can be expressed with the help of mappings f_z and f_y as follows:

$$\begin{array}{ll}
f_z(X^i, y^k) \Rightarrow Z^a & f_y(X^i, y^s) \Rightarrow Y^k \\
f_z(X^j, y^k) \Rightarrow Z^b & f_y(X^j, y^k) \Rightarrow Y^t \\
f_z(X^j, y^t) \Rightarrow Z^b & f_y(Z^n, y^t) \Rightarrow Y^p \\
f_z(X^n, y^t) \Rightarrow Z^c & f_y(X^q, y^p) \Rightarrow Y^u \\
f_z(X^n, y^p) \Rightarrow Z^c & f_y(X^h, y^u) \Rightarrow Y^r \\
f_z(X^q, y^p) \Rightarrow Z^d & \\
f_z(X^q, y^u) \Rightarrow Z^d & \\
f_z(X^h, y^u) \Rightarrow Z^e, &
\end{array}$$

or using the differential mappings:

$$\begin{array}{ll}
\varphi_z(B_i^j, y^k) \Rightarrow K_a^b & \varphi_y(B_i^j, y^k) \Rightarrow A_k^t \\
\varphi_z(B_j^n, y^t) \Rightarrow K_b^c & \varphi_y(B_j^n, y^t) \Rightarrow A_t^p \\
\varphi_z(B_n^q, y^p) \Rightarrow K_c^d & \varphi_y(B_n^q, y^p) \Rightarrow A_p^u \\
\varphi_z(B_q^h, y^u) \Rightarrow K_d^e & \varphi_y(B_q^h, y^u) \Rightarrow A_u^r
\end{array}$$

The fragment of the classical flow table [1] in Fig. 4 is filled out according to the specification prescribed by the sequence given above. The symbol "$-$" emphasises that no prescription for Z or Y can be derived from the sequences. The continuous arrows show state transitions corresponding to the synchronous mode.

Along the dashed arrows, a normal asynchronous mode is illustrated by properly fixed values of Y-s not prescribed in the original sequences.

There are control procedures in which certain prescribed input changes must not cause any change in the output and secondary combination. For example, let a fragment of the corresponding prescribed sequences be given as follows:

$$\begin{array}{l}
\ldots X^1\ X^2\ X^4\ X^3\ X^5\ X^6\ X^2\ X^7\ X^8\ X^9 \ldots \\
\ldots Y^1\ Y^3\ Y^2\ Y^2\ Y^2\ Y^2\ Y^4\ Y^6\ Y^6\ Y^7 \ldots \\
\ldots Z^1\ Z^2\ Z^5\ Z^5\ Z^5\ Z^5\ Z^3\ Z^3\ Z^4\ Z^1 \ldots
\end{array}$$

The indices only serve to distinguish the combinations from one another of own kind. So, for example, X^2, Y^2 and Z^2 do not represent necessarily the same binary combination.

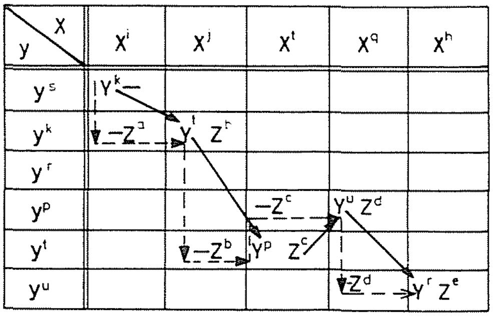


Fig. 4

The control unit specified by the above fragment of prescribed sequences can be considered as insensitive to the fragment X^4, X^3, X^5, X^6 occurring after X^2, because no changes are prescribed either in the secondary or in the output combination.

Using the notation for the changes of combinations:

$$\ldots B_1^2\ B_2^4\ B_4^3\ B_3^5\ B_5^6\ B_6^2\ B_2^7\ B_7^8\ B_8^9\ \ldots$$
$$\ldots A_1^3\ A_3^2\ A_2\ A_2\ A_2\ A_2^4\ A_4^6\ A_6\ A_6^7\ \ldots$$
$$\ldots K_1^2\ K_2^5\ K_5\ K_5\ K_5\ K_5^3\ K_3\ K_3^4\ K_4^1$$

where $\begin{Bmatrix} A_i \\ K_i \end{Bmatrix}$ denotes *fictive* $\begin{Bmatrix} \text{secondary} \\ \text{output} \end{Bmatrix}$ change, during which the $\begin{Bmatrix} \text{secondary} \\ \text{output} \end{Bmatrix}$ combination preserves the value $\begin{Bmatrix} Y^i \\ Z^i \end{Bmatrix}$.

A more concise description can be formed by sectioning the prescribed input changes according to the fragments of insensitivity. A control procedure remains unaltered and corresponds to the original specification if the control unit is defined to be insensitive to any sequences of input combinations or changes occurring between the beginning and the end of a fragment of insensitivitiy.

In this sense, the changes from the beginning to the end of a fragment of insensitivity are not considered to be prescribed ones. In the example mentioned above, this interpretation means that the control unit is prescribed to be insensitive not only to the sequence X^4, X^3, X^5, X^6, but to any other sequences of prescribed input combinations not containing X^2. In this way, the original specification of the control procedure becomes more rigorous, because the insensitivity is also prescribed for some of those input sequences whose effects were unspecified originally.

This modification should be useful because it gives a more concise description.

Let $[B_i^j]$ denote *the prescribed input section change* which represents all of the possible sequences of prescribed input changes starting with X^i ending with X^j and containing X^j only once (at the end).

Introducing the notation of the prescribed input section change, the fragment in the example can be rewritten in a more concise form as follows:

$$\begin{array}{l} \ldots\ B_1^2\ B_2^4\ [B_4^2]\ B_2^7\ B_7^8\ B_8^9 \\ \ldots\ A_1^3\ A_3^2\ A_2^4\ A_4^6\ A_6\ A_6^7 \\ \ldots\ K_1^2\ K_2^5\ K_5^3\ K_3\ K_3^4\ K_4^1 \end{array}$$

In the flow-chart description of synchronous control units, we may usually define output pulses as two prescribed output changes occurring immediately after each other without any prescribed input changes. As an example of this, let the fragment of combination sequences be given as follows:

$$\begin{array}{l} \ldots\ X^1\ X^3\ X^4\ \quad\ X^5\ X^6\ \ldots \\ \ldots\ Y^1\ Y^2\ Y^3\ Y^4\ Y^1\ Y^5\ \ldots \\ \ldots\ Z^1\ Z^4\ Z^5\ Z^1\ Z^3\ Z^6\ \ldots \end{array}$$

As a response to X^4, Y^3 and Z^5 occur. Without any further input changes the secondary and output combination change to Y^4 and Z^1. The duration of Y^3 and Z^5 depends on the hardware structure and this duration in turn determines the width of the pulse.

In the formal description based on the prescribed changes, the *fictive input change* can be introduced:

$$\begin{array}{l} \ldots\ B_1^3\ B_3^4\ B_4\ B_4^5\ B_5^6\ \ldots \\ \ldots\ A_1^2\ A_2^3\ A_3^4\ A_4^1\ A_1^5\ \ldots \\ \ldots\ K_1^4\ K_4^5\ K_5^1\ K_1^3\ K_3^6\ \ldots \end{array}$$

The notation B_i represents formally the fictive input change, during which the input combination X^i remains at the input.

II. The specification of the control procedure by the prescribed input and output changes

Let $B(K_n^m)$ denote the set of all prescribed input changes which cause the output change K_n^m in at least one prescribed sequences of input and output changes.

Let $\left\{\begin{matrix} K(K_n^m) \\ (K_n^m)K \end{matrix}\right\}$ denote the set of all prescribed output changes which are adjacent to K_n^m as $\left\{\begin{matrix} \text{predecessors} \\ \text{successors} \end{matrix}\right\}$ in at least one prescribed sequence of output changes.

Let $(K_n^m)B$ denote the set of all prescribed input changes which are adjacent as successors to at least one element of $B(K_n^m)$ in at least one prescribed sequence of input changes.

The meaning of the notations $B(K_n^m)$, $K(K_n^m)$, $(K_n^m)K$ and $(K_n^m)B$ can be formally extended also for fictive changes.

Let $\left\{\begin{matrix} B(K) \\ K(K) \\ (K)K \\ (K)B \end{matrix}\right\}$ denote the set of all sets $\left\{\begin{matrix} B(K_n^m) \\ K(K_n^m) \\ (K_n^m)K \\ (K_n^m)B \end{matrix}\right\}$ specified in the control procedure for every $K_n^m \in K$ including also the fictive changes. Let $B:K$ denote the set of the sets $B(K)$, $K(K)$, $(K)K$ and $(K)B$ specified in the control procedure.

Theorem 1

The set $B:K$ is an unambiguous representation of a control procedure specified by the prescribed sequences of input and output changes, if and only if no such K_n^m, K_m^p and K_m^r can be found, for which

$$K_m^p \in (K_n^m)K \qquad \text{and} \qquad K_m^r \in (K_n^m)K$$

and $(K_n^m)B \cap B(K_m^p) \cap B(K_m^r) \neq \emptyset$ are valid.

The notations $\in$ and $\cap$ have the meaning as used in the algebra of sets and $\emptyset$ denotes the empty set.

Proof

Necessity

If there exists a B_i^j, such that $(K_n^m)B \cap B(K_m^p) \cap B(K_m^r) = B_i^j$, then it cannot be said, whether K_m^p or K_m^r would be the response to B_i^j occurring after K_n^m.

Sufficiency

If the expression $(K_n^m)B \cap B(K_m^p) \cap B(K_m^r) = \emptyset$ is valid for every $K_m^p \in (K_n^m)K$ and $K_m^r \in (K_n^m)K$, then the response can be uniquely predicted for every prescribed input change occurring after each output change according to the initial specification.

As an example for Theorem 1, let us consider firstly a control procedure specified by the prescribed sequences of input and output changes as follows:

$$\begin{array}{ccccccc} \overset{\downarrow}{B}{}_1^3 & B_3^2 & B_2^4 & B_4^2 & B_2^4 & B_4^2 & B_2^1 \\ K_1^4 & K_4 & K_4 & K_4 & K_4 & K_4 & K_4^1 \end{array}$$

The arrow at the beginning shows the starting point, i.e. the initial state. Let us assume that finishing a prescribed sequence means always returning to the initial state.

The set $B:K$ can be generated from the sequences of input and output changes:

$$\begin{array}{lll} B(K_1^4) = \{B_1^3\}; & B(K_4) = \{B_4^2, B_2^4, B_3^2\}; & B(K_4^1) = \{B_2^1\} \\ K(_1^4) = \{K_4^1\}; & K(K_4) = \{K_4, K_1^4\}; & K(K_4^1) = \{K_1^4\} \\ (K_1^4)K = \{K_4\}; & (K_4)K = \{K_4, K_4^1\}; & (K_4^1)K = \{K_1^4\} \\ (K_1^4)B = \{B_3^2\}; & (K_4)B = \{B_2^4, B_4^2, B_2^1\}; & (K_4^1)B = \{B_1^3\} \end{array}$$

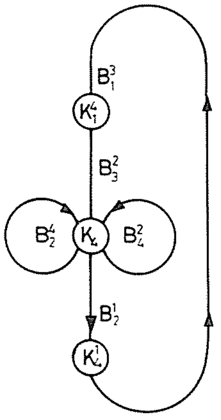
In Figure 5.a, the set $B:K$ is represented by a table *called the $B:K$ table.* The rows of the $B:K$ table contain the elements of the sets K, $B(K)$, $K(K)$, $K(K)$, $(K)B$ respectively. In a cell of the rows, the corresponding input or output changes are characterised only by their lower and upper indices. So, for example, 32 is written instead of B_3^2.

In Figure 5.b a directed graph representation is shown for the set $B:K$. The nodes of the graph represent the output changes and the edges are defined by the input changes. For flowchart-like interpretation let this graph be drawn in the form shown in Fig. 5.c. Let this form of the directed graph generated by a $B:K$ table be called the *$B:K$ graph.*

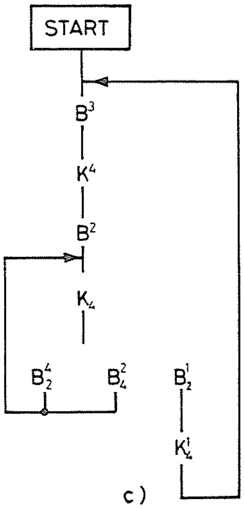
The $B:K$ set of this example does not contain any pairs of output changes which would exclude the unambiguous representation of the prescribed sequences according to Theorem 1. It can be observed in Fig. 5 that the prescribed sequences given above can be read out from the $B:K$ table or $B:K$ graph. In this case not only the sequences initially prescribed are represented by the set $B:K$, but the number of the represented sequences can be considered as infinitely large. Namely, all sequences, in which the series B_2^4, B_4^2 with K_4

K	14	4	41
B(K)	13	42 24 32	21
K(K)	41	4 14	4
(K) K	4	4 41	14
(K) B	32	24 42 21	13

a)



b)



c)

Fig. 5

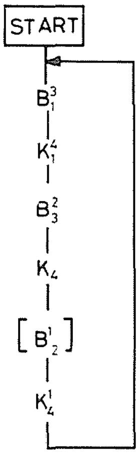
repeats itself several times, are included in the set $B:K$. For that reason, in this example, a more concise initial specification can be formed by using the possibility of defining prescribed input section changes as follows:

$$\overset{\downarrow}{B}{}_1^3 \; B_3^2 \; [B_2^1]$$
$$K_1^4 \; K_4 \; K_4^1$$

The $B:K$ table and graph are shown in Fig. 6. It is obvious that in this case the set $B:K$ represents not only the repeating series $B_2^4 B_4^2$ with K_4 but also every sequence in which there occurs any input combination preserving Z^4 after X^2 with the exception of X^1.

K	14	4	41
B(K)	13	32	[21]
K(K)	41	14	4
(K)K	4	41	14
(K)B	32	21	13

Fig. 6



K	14	4	42	21
B(K)	13	32 24 42	24	41
K(K)	21	4 14	4	42
(K)K	4	4 42	21	14
(K)B	32	24 42	41	13

a)

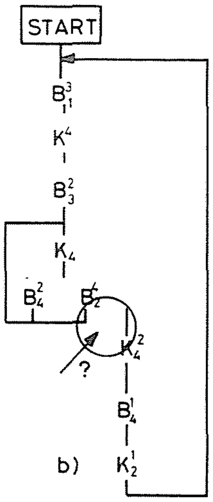


Fig. 7

As an other example for Theorem 1, let a control procedure be given by the prescribed sequences as follows:

$$\begin{array}{cccccccc} \overset{\downarrow}{B}{}_1^3 & B_3^2 & B_2^4 & B_4^2 & B_2^4 & B_4^2 & B_2^4 & B_4^1 \\ K_1^1 & K_4 & K_4 & K_4 & K_4 & K_4 & K_4^2 & K_2^1 \end{array}$$

From the $B{:}K$ table shown in Fig. 7.a, it can be seen that

$$K_4^2 \in (K_4)K; \quad K_4 \in (K_4)K; \quad (K_4)B \cap B(K_4^2) \cap B(K_4) = \{B_2^4\}$$

Thus, according to Theorem 1, the set $B{:}K$ cannot represent unambiguously the sequences given initially. The problem emerges, when we try to construct the $B{:}K$ graph. This is illustrated in Fig. 7.b. There is no information in the set $B{:}K$ as to whether K_4 repeats itself or is followed by K_4^2 at the occurrence of B_2^4.

It is obvious that a counting-like procedure, as given in the example, yields always ambiguous $B{:}K$ sets.

To eliminate this problem, let the *recurrent output changes* be introduced.

Let $Q_1 \ldots Q_i \ldots Q_4$ be proper nonempty disjoint subsets of

$$\left\{ \begin{array}{l} B(K_n^m) \\ K(K_n^m) \\ (K_n^m)K \\ (K_n^m)B \end{array} \right\}.$$

K_n^m can be substituted by its copies as *recurrent changes* $K_n^m/1, \ldots K_n^m/i, \ldots K_n^m/h$, for which

$$\left\{ \begin{array}{l} B(K_n^m/i) = Q_i \\ K(K_n^m/i) = Q_i \\ (K_n^m/i)K = Q_i \\ (K_n^m/i)B = Q_i \end{array} \right\} \qquad (1 \le i \le h).$$

The sets

$$\left\{ \begin{array}{l} K(K_n^m/i),\ (K_n^m/i)K,\ (K_n^m/i)B \\ B(K_n^m/i),\ (K_n^m/i)K,\ (K_n^m/i)B \\ B(K_n^m/i),\ K(K_n^m/i),\ (K_n^m/i)B \\ B(K_n^m/i),\ K(K_n^m/i),\ (K_n^m/i)K \end{array} \right\}$$

can be determined formally, based on the set $B:K$, following the consequences of splitting the sets according to Q_i-s.

Let $K_4/1$ and $K_4/2$ be introduced instead of K_4 in the example shown in Fig. 7 according to the subsets of $B(K_4): Q_1=\{B_3^2B_2^4\}$, $Q_2=\{B_4^2\}$.

The modified $B:K$ set is shown in Fig. 8. However, it is not unambiguous at this point, because $K_4/1 \in (K_4/2)K$; $K_4^2 \in (K_4/2)K$ and $(K_4/2)B \cap B(K_4/1) \cap B(K_4/2)=\{B_2^4\}$.

K	14	4/1	4/2	42	21
$B(K)$	13	32 24	42	24	41
$K(K)$	21	4/2 4/1 14	4/1	4/2	42
$(K)K$	4/1	4/1 4/2	42 4/1	21	14
$(K)B$	32	24 42	24	41	13

Fig. 8

Now, let $K_4/11$ and $K_4/12$ be introduced instead of $K_4/1$ according to the subsets of $K(K_4/1)$: $Q_1=\{K_1^4, K_4/1\}$, $Q_2=\{K_4/2\}$.

The modified $B:K$ set in Fig. 9 shows that

$$K_4/12 \in (K_4/2)K; \quad K_4^2 \in (K_4/2)K \quad \text{and}$$

$$(K_4/2)B \cap B(K_4/12) \cap B(K_4^2)=\{B_2^4\}$$

So, even after this, the $B:K$ set is not unambiguous.

K	14	4/11	4/12	4/2	42	21
$B(K)$	13	32 24	24	42	24	41
$K(K)$	21	4/11 14	4/2	4/12 4/11	4/2	42
$(K)K$	4/11	4/11	4/2	4/12 42	21	14
$(K)B$	32	24 42	42	24	41	13

Fig. 9

Let $K_4/21$ and $K_4/22$ be introduced instead of $K_4/2$ according to the subsets of $K(K_4/2)$: $Q_1=\{K_4/11\}$, $Q_2=\{K_4/12\}$. Finally, this modification, shown in Fig. 10, yields an unambiguous $B:K$ set according to Theorem 1.

K	14	4/11	4/12	4/21	4/22	42	21
$B(K)$	13	32 24	24	42	42	24	41
$K(K)$	21	4/11 14	4/21	4/11	4/12	4/22	42
$(K)K$	4/11	4/11 4/21	4/22	4/12	42	21	14
$(K)B$	32	24 42	42	24	24	41	13

Fig. 10

The recurrent output changes introduced above result in the notations in the prescribed sequences as follows:

$$\begin{array}{cccccccc} \overset{\downarrow}{B}{}_1^3 & B_3^2 & B_2^4 & B_4^2 & B_2^4 & B_4^2 & B_2^4 & B_4^1 \\ K_1^4 & K_4/11 & K_4/11 & K_4/21 & K_4/12 & K_4/22 & K_4^2 & K_2^1 \end{array}$$

The $B{:}K$ graph derived from Fig. 10 is shown in Fig. 11. The unambiguous $B{:}K$ set is always derivable by the steps illustrated above. In the case of counting-like control procedures, the number of steps needed to introduce recurrent output changes can be greatly reduced by using counters as functional units outside of the control unit.

If the set $B:K$ is given, then the control procedure is specified, because the prescribed sequence of output changes is determined by the set $B:K$ for every prescribed sequence of input changes. For, it is obvious from its definition that the set $B{:}K$ represents all of the prescribed sequences of input and output changes and also the correspondence between them.

Later in this paper, it will be assumed that the control procedure is specified by the unambiguous $B{:}K$ set without fixing the secondary combinations and their changes. The secondary combinations will be defined during the synthesis procedure as the states and the state transitions of the control unit.

It can be shown that the sets K and $B(K)$ contain all the necessary information for the construction of a $B{:}K$ set which will represent all of the possible sequences of input and output changes, provided that these changes are not excluded by K and $B(K)$. Let this type of set $B:K$ be called *completely calculated* $B{:}K$ *set.*

The control procedure may have extra restrictions on the prescribed sequences excluding some of total number of possible sequences calculatable from the sets K and $B(K)$. In this case, an *incompletely calculated* $B{:}K$ *set* is obtained.

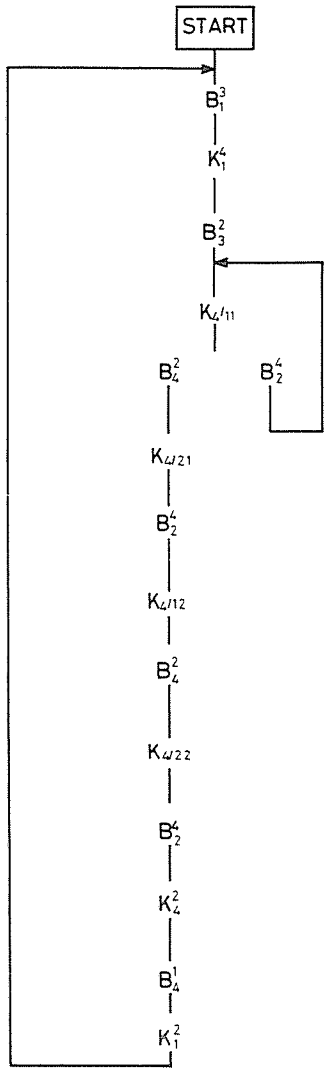


Fig. 11

If the whole set $B:K$ is given as the initial specification of the control procedure, then it will be called the *initially defined $B:K$ set.*

The calculation of the set $B:K$ can be based on the operations and rules as follows:

Let the operation denoted by :d be defined on a set

$$\left\{\begin{matrix}\{B_a^b, [B_c^d] \dots B_i^j, B_j, B_p, B_p^n\} \\ \{K_a^b, K_c^d \dots K_n^m, K_n, K_h, K_s^t\}\end{matrix}\right\} \quad \text{of}$$

$$\begin{Bmatrix}\text{input}\\ \text{output}\end{Bmatrix} \text{ changes with the resulting set}$$

$$\begin{Bmatrix}\{X^b, X^d \dots X^j, X^p, X^n\}\\ \{Z^b, Z^d \dots Z^m, Z^n, Z^h, Z^t\}\end{Bmatrix} \text{ of } \begin{Bmatrix}\text{input}\\ \text{output}\end{Bmatrix} \text{ combinations.}$$

Let the operation denoted by $^{-1}$ be defined on a set

$$\begin{Bmatrix}\{B^b_a, [B^d_c] \dots B^j_i, B_j, B^n_p\}\\ \{K^b_a, K^d_c, \dots K^m_n, K_n, K_h, K^t_s\}\end{Bmatrix} \text{ of}$$

$$\begin{Bmatrix}\text{input}\\ \text{output}\end{Bmatrix} \text{ changes with the resulting set}$$

$$\begin{Bmatrix}\{B^a_b, [B^c_d] \dots B^i_j, B_j, B_r, B^p_n\}\\ \{K^a_b, K^c_d, \dots K^n_m, K_n, K_h, K^s_t\}\end{Bmatrix} \text{ of } \begin{Bmatrix}\text{input}\\ \text{output}\end{Bmatrix} \text{ changes.}$$

According to their definitions, the operations $:d$ and $^{-1}$ handle the fictive changes in a special way.

Rule 1: If $K^m_n \in K(K^p_m)$, then

$$B(K^m_n):d \cap B(K^p_m)^{-1}:d \neq \emptyset \quad \text{and} \quad (K^m_n)B \cap B(K^p_m) \neq \emptyset$$

Rule 2: If $K^r_p \in (K^p_m)K$, then

$$B(K^p_m):d \cap B(K^r_p)^{-1}:d \neq \emptyset \quad \text{and} \quad (K^p_m)B \cap B(K^r_p) \neq \emptyset$$

Rule 3: If $B^j_i \in (K^m_n)B$, then $B(K^m_n) \cap B^i_- \neq \emptyset$ and there exists a K^r_m, such that $K^m_n \in K(K^r_m)$ and $B^j_i \in B(K^r_m)$ are valid (B^i_- denotes the set of all input changes finished by X^i)

Rule 4: If $B^j_i \in B(K^m_n)$, then $(K^m_n)B \cap B^-_j \neq \emptyset$ and there exists a K^n_p, such that $K^n_p \in K(K^m_n)$ and $B^j_i \in (K^n_p)B$ are valid. (B^-_j denotes the set of all input changes starting from X^j)

The rules summarised above are also valid and formally extendable for fictive and recurrent changes. Let the output signals be called *affected* by the output changes K^m_n if their values alter during K^m_n.

Later in this paper, it is assumed that the fixed hardware structure of the control unit stores not only the secondary combinations as states, but also the output combinations by output flip-flops. This extra function is usual in practice, since it also prevents output hazards as is shown in the synchronous phase register structure in Fig. 2.

In this case, the prescribed output and secondary changes can be uniquely characterised by fixing only the initial output and secondary combinations and by giving step by step those output and secondary variables which correspond to the changing of the affected output and secondary signals.

Thus, the mappings φ_z and φ_y can be substituted by mappings which result in changes of output and secondary signals instead of changes of combinations.

Let

$$\varphi_z(B_i^j, y^k) \Rightarrow K_n^m \quad \text{and} \quad \varphi_y(B_i^j, y^k) \Rightarrow A_k^p$$

be given assuming that

$$K_n^m : Z_r \bar{Z}_s Z_v \quad \text{and} \quad A_k^p : \bar{Y}_a Y_b \bar{Y}_c$$

which represents that Z_r, Z_v and Y_b change from 0 to 1, Z_s, Y_a and Y_c change from 1 to 0 during K_n^m and A_k^p respectively.

Let the notations be introduced as follows:

$$P_n^m = ((Z_r Z_v)(Z_s)) \quad S_k^p = ((Y_b)(Y_a Y_c)),$$

where $\begin{Bmatrix} P_n^m \\ S_k^p \end{Bmatrix}$ is an incompletely specified two-block-partition on the set of the $\begin{Bmatrix} \text{output} \\ \text{secondary} \end{Bmatrix}$ variables corresponding to the $\begin{Bmatrix} \text{output} \\ \text{secondary} \end{Bmatrix}$ signals. The first block of $\begin{Bmatrix} P_n^m \\ S_k^p \end{Bmatrix}$ contains the variables changing from 0 to 1 during $\begin{Bmatrix} K_n^m \\ A_k^p \end{Bmatrix}$. The second block of $\begin{Bmatrix} P_n^m \\ S_k^p \end{Bmatrix}$ contains the variables changing 1 to 0 during $\begin{Bmatrix} K_n^m \\ A_p^k \end{Bmatrix}$.

Let the set of all $\begin{Bmatrix} P_n^m \\ S_k^p \end{Bmatrix}$ be called the set of $\begin{Bmatrix} P \\ S \end{Bmatrix}$ partitions. Using this notations, the mappings V_z and V_y can be defined as follows:

$$V_z(B_i^j, y^k) \Rightarrow P_n^m, \qquad V_y(B_i^j, y^k) \Rightarrow S_k^p,$$

or in general:

$$V_z(B, y) \Rightarrow P \qquad V_y(B, y) \Rightarrow S$$

In consequence of having introduced the fictive changes, the result of the mapping $\begin{Bmatrix} V_z \\ V_y \end{Bmatrix}$ can also mean fictive changes. In such cases, empty partitions denoted by $\begin{Bmatrix} P_n \\ S_k \end{Bmatrix}$ are used for formal interpretation.

Let $S(K_n^m)$ denote the S partition belonging to K_n^m. Let $A(K_n^m)$ denote the prescribed secondary change belonging to K_n^m.

A secondary change belongs to K_n^m if it is generated during K_n^m in every prescribed sequence of changes containing K_n^m.

III. The compatibility of the prescribed output changes

Let a $B:K$ set or $B:K$ graph be called an *open* one, if it does not contain any output changes K_n^m, such that

$$B(K_n^m) \cap B(K_n^m)^{-1} \neq \emptyset$$

It is obvious, that any unambiguous $B:K$ set or $B:K$ graph can be made open by introducing recurrent output changes. For example, a $B:K$ graph is given in Fig. 12. This graph is not open, because

$$B(K_3^4) \cap B(K_3^4)^{-1} = \{B_3^4\} \neq \emptyset .$$

Introducing $K_3^4/1$ and $K_3^4/2$ according to $Q_1 = B_3^4$ and $Q_2 = B_4^3$ of $B(K_3^4)$, the $B:K$ graph becomes open, as it is shown in Fig. 13. Later on, it is assumed that the $B:K$ sets and $B:K$ graphs are open and unambiguous.

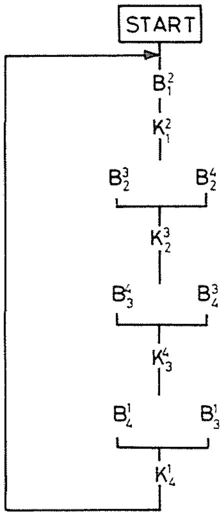


Fig. 12

Two prescribed output changes K_1^m and K_p^r may occur *in the same state* y^k if

$$V_z(B \in B(K_n^m), y^k) \Rightarrow P_n^m \quad \text{and}$$
$$V_z(B \in B(K_p^r), y^k) \Rightarrow P_p^r \quad \text{and}$$
$$V_y(B \in B(K_n^m), y^k) \Rightarrow S(K_n^m) \quad \text{and}$$
$$V_y(B \in B(K_p^r), y^k) \Rightarrow S(K_p^r) \quad \text{are valid.}$$

In other words, both P_n^m, P_p^r and $S(K_n^m)$, $S(K_p^r)$ can be distinguished by the input changes in the common state y^k.

Let the output changes K_n^m and K_p^r be said to be *parallel* if every output signal affected by either K_n^m or K_p^r has the same value both in Z^m and Z^r. In notation: $K_n^m \| K_p^r$. For example, if

	Z_1	Z_2	Z_3	Z_4	Z_5
Z^n:	0	0	1	0	1
Z^m:	1	1	0	0	1
Z^p:	0	1	1	1	0
Z^r:	1	1	0	1	1

then $K_n^m \| K_p^r$, because $K_n^m : Z_1 Z_2 \bar{Z}_3$, $K_p^r : Z_1 \bar{Z}_3 Z_5$ and $Z_5 = 1$ in Z^m and $Z_2 = 1$ in Z^r.

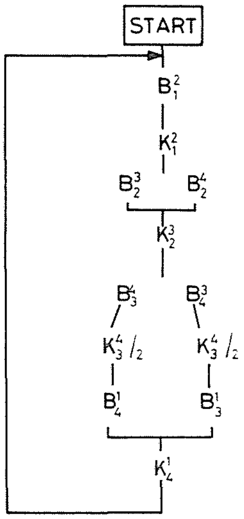


Fig. 13

For two arbitrary fictive output changes, $K_n \| K_p$ is always true, because fictive output changes do not affect any output signals.

The relation $\|$ can be interpreted also for secondary changes. If the hardware structure constrains the secondary combinations to be only in 1-from-n-code, then $A(K_n^m) \| A(K_p^r)$ can be valid only if both $A(K_n^m)$ and $A(K_p^r)$ are fictive or $A(K_n^m) = A(K_p^r)$.

Theorem 2

Assuming that the relations $K_n^m \| K_p^r$, $K_n^m \| K_p$, $K_p^r \| K_n$ are not valid, the prescribed output changes K_n^m and K_p^r may occur in the same state if and only if

$$B(K_n^m):d \cap B(K_p^r):d = \emptyset \quad \text{and}$$

$$B(K_n^m):d \cap B(K_p^r)^{-1}:d = \emptyset \quad \text{and}$$

$$B(K_n^m)^{-1}:d \cap B(K_p^r):d = \emptyset$$

are valid.

Proof

Necessity

If any of the expressions in the theorem were not valid, then input changes might occur in a common state which would cause the changes according to both P_n^m and P_p^r.

Sufficiency

If the expressions of the theorem are valid, then in the common state y^k, no such prescribed input combinations can be present or occur which would cause the changes according to both P_n^m and P_p^r. So, the mapping V_z is able to distinguish P_n^m from P_p^r based on the prescribed input changes only, assuming the secondary combination y^k. Similarly, the mapping V_y is also able to distinguish S_n^m from S_p^r with the help of the prescribed input changes in y^k.

It can be proved that if two prescribed output changes may occur in the same state, then they are in a compatibility relationship. In notation: $K_n^m \sim K_p^r$.

In special cases, the conditions of Theorem 2 can be relaxed independently of each other. For example, it can be proved [10] that if $K_n^m \| K_u^v$, $K_n^m \| K_u$, $K_u^v \| K_n$ are true simultaneously and both $A(K_n^m)$ and $A(K_u^v)$ are fictive, then $K_n^m \sim K_u^v$ is fulfilled independently of the $B(K)$ set. So, this case can be called *input-independent compatibility*. It can also be shown [10] that if $K_n^m \| K_u^v$, $K_n^m \| K_u$ are true and both $A(K_n^m)$ and $A(K_u^v)$ are fictive, then the condition

$$B(K_n^m)^{-1}:d \cap B(K_u^v):d = \emptyset$$

is sufficient and necessary for $K_n^m \sim K_u^v$.

Let another example be mentioned for the case of $K_n^m \in K(K_m^n)$ and $K_m^n \in (K_n^m)$. In this case, the sufficient and necessary condition for $K_n^m \sim K_u^v$ is

$$B(K_n^m):d \cap B(K_u^v):d = \emptyset$$

only [10].

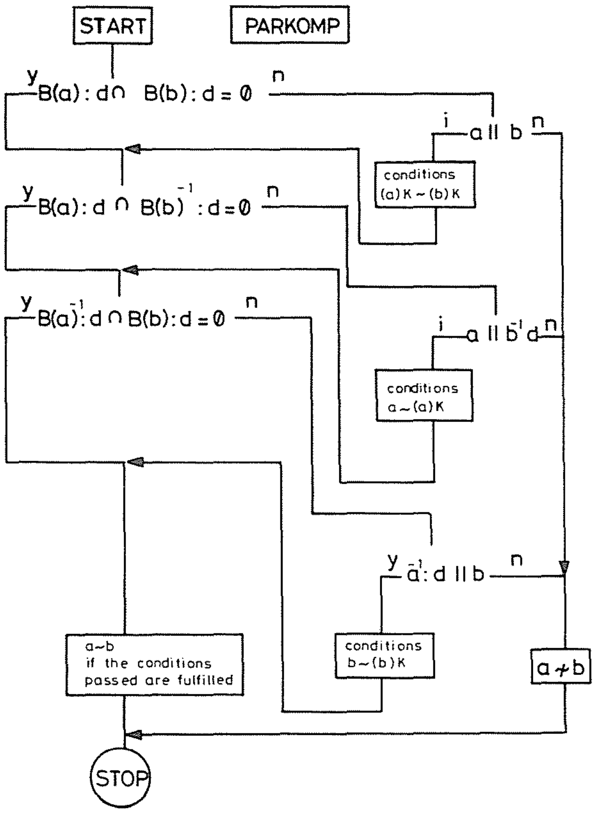


Fig. 14

Based on Theorem 2 and considering all of the special cases [10], an algorythm can be developed for the pair-wise compatibility checking of the prescribed output changes. This algorythm called PARKOMP is illustrated by a flow chart in Fig. 14. The output changes to be checked are denoted by a and b.

The notation $a^{-1}:d$ represents formally a fictive output change which can be derived from a in the case of $a=K_n^m$, as K_n. The algorythm PARKOMP may result in a conditional compatibility relationship between a and b. The conditions arise at three points of the flow chart, where they are represented by $(a)K\sim(b)K$, $a\sim(a)K$, $b\sim(b)K$, respectively.

The meaning of these symbols is as follows:

— $(a)K\sim(b)K$ means the condition which is fulfilled, when each successor of a is compatible with each successor of b.

— $a\sim(a)K$ means the conditions which are fulfilled, when a is compatible with each of its successors.

— $b \sim (b)K$ means the conditions which are fulfilled, when b is compatible with each of its successors.

The algorythm PARKOMP provides a conditional compatibility if the termination $a \sim b$ in the flow chart of Fig. 14 is reached by passing at least one of the possible conditions. It is assumed in Fig. 14 that the secondary combinations are in 1-from-n code.

With the help of the algorythm PARKOMP, the well-known compatibility table can be filled out and the maximal compatibility classes of the prescribed output changes can be determined.

Based on these compatibility classes, special closed covers are obtainable for the state-definition.

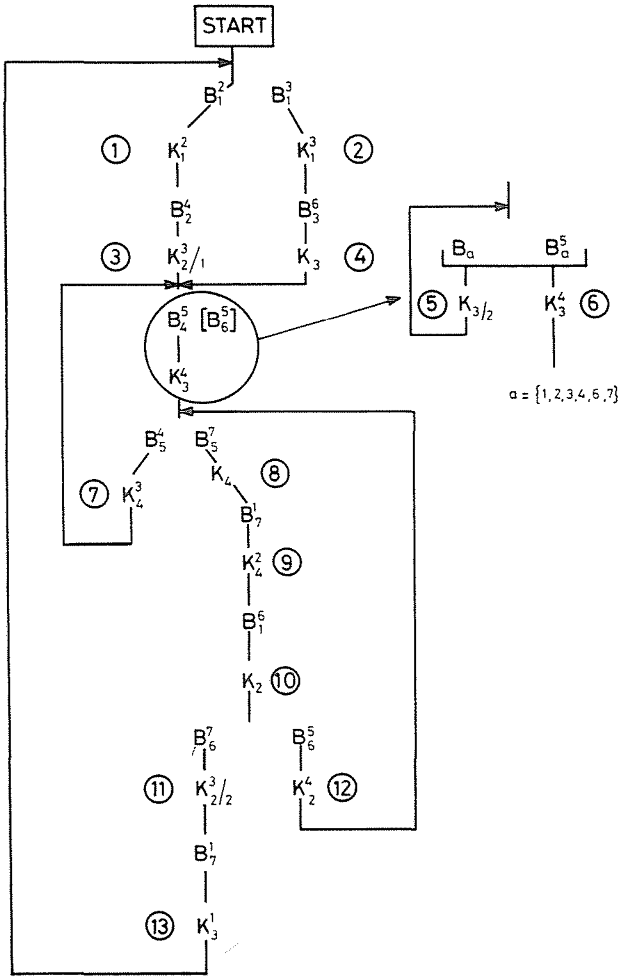


Fig. 15

As an example let a $B{:}K$ graph be given in Fig. 15. The formal transformation of the circled part is made for the simple handling of the input section change during the algorythm PARKOMP [10]. The $B:K$ set is shown in Fig. 16. Let the prescribed output combinations be assumed as shown in Fig.

K	12	13	23/1	3	34	43	4	42	2	23/2	24	31
$B(K)$	12	13	24	36	45 [65]	54	57	71	16	67	65	71
$K(K)$	31	31	12	13	23/1 3,34	34	34 24	4	42	2	2	23/2
$(K)K$	23/1	3	34	34	43 4	34	42	2	23/2 24	31	43 4	12 13
$(K)B$	24	36	45 [65]	45 [65]	54 57	45 [65]	71	16	67 65	71	54 57	12 13

Fig. 16

	Z_1	Z_2	Z_3	Z_4
Z^1	0	1	0	0
Z^2	0	0	0	1
Z^3	0	0	1	0
Z^4	1	0	0	1

	Z_1	Z_2	Z_3	Z_4
K_1^2	0	0	0	1
	0	1	0	0
K_1^3	0	0	1	0
	0	1	0	0
K_2^3	0	0	1	0
	0	0	0	1
K_3^4	1	0	0	1
	0	0	1	0
K_4^3	0	0	1	0
	1	0	0	1
K_4^2	0	0	0	1
	1	0	0	1
K_2^4	1	0	0	1
	0	0	0	1
K_3^1	0	1	0	0
	0	0	1	0

Fig. 17

17, where also the real (not fictive) prescribed output changes are illustrated. A computer program [10] applying pair-wise the algorythm PARKOMP constructs the compatibility table and provides the maximal compatibility classes as follows:

(1, 2, 3, 4, 7, 8, 9, 12) (1, 2, 3, 4, 7, 8, 12, 13)
(1, 2, 3, 4, 7, 11, 12, 13) (1, 2, 3, 7, 8, 9, 10, 12)
(1, 2, 3, 7, 10, 11, 12) (2, 3, 4, 5, 6, 7, 12)
(2, 3, 4, 6, 7, 8, 9, 12) (2, 3, 4, 6, 7, 11, 12)
(2, 3, 6, 7, 8, 9, 10, 12) (2, 3, 6, 7, 10, 11, 12)

The correspondence between the numbers and the output changes is shown in Fig. 15.

IV. The definition of the states

Every state of a control unit specified by $B:K$ set or graph corresponds uniquely to a compatibility class of the prescribed output changes. These classes are disjoint and realise a closed cover of the prescribed output changes. The proof of the above statement [10] can be derived from the requirement that the mappings V_z and V_y should distinguish the P and S sets of the prescribed output changes belonging to the same compatibility class with the help of the prescribed input changes only.

Let the set of such compatibility classes be called a *state-defining partition* of the prescribed output changes.

For example, a state-defining partition is in the case of the above maximal compatibility classes as follows:

$$\pi = \{(K_1^2\, K_1^3\, K_2^3/1 K_2^3/2 K_1^3\, K_2^4)\,(K_4^3\, K_4 K_2)\,(K_3 K_3^4)\}$$

The three blocks define three states which will be denoted by the secondary combinations y^1, y^2, y^3 respectively. Thus, in the example of Fig. 15.:

A_1^3 belongs to $K_2^3/1$ and K_1^3
A_2^3 belongs to K_4^3
A_1^2 belongs to K_2^4
A_3^2 belongs to K_3^4
A_2^1 belongs to K_2

The $B:K$ graph of Fig. 15 is drawn in Fig. 18 completed with the secondary changes. The dashed lines separate the values of the secondary

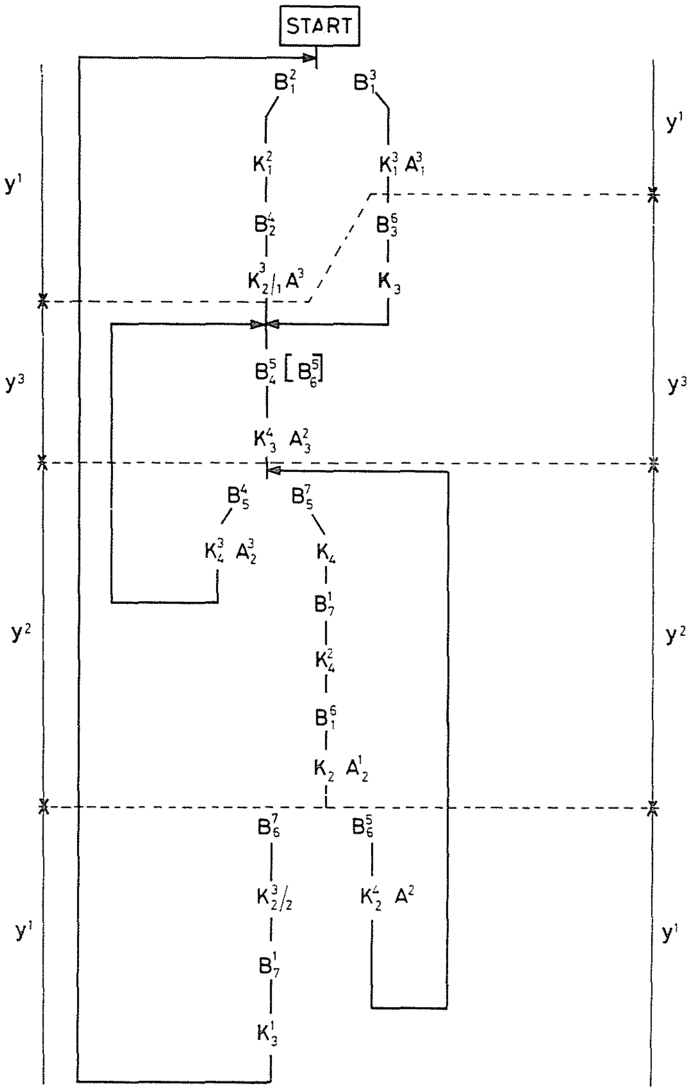


Fig. 18

combinations y^1, y^2, y^3. In the case of a phase-register structure, the dashed lines define the phases.

It is obvious that in general, more than one state-defining partition may be found for a control unit. A state-defining partition can be considered as an optimal one, if it defines the smallest number of states and state-transitions. Before looking for the optimal solution, another property of a state-defining partition is to be considered.

Let the prescribed output change K_n^m be called a *junction-in* output change if there exist the output changes $K_m^p, K_m^r \dots K_m^t$ for which $K_n^m \in K(K_m^p)$, $K_n^m \in K(K_m^r), \dots K_n^m \in K(K_m^t)$ are true. In this case, let $K_m^p, K_m^r, \dots K_m^t$ be called the *junction-out* output changes belonging to K_n^m.

All junction-out output changes belonging to the same junction-in output are in a common block of a state-defining partition. So, if a junction-in output change is incompatible with at least one of its junction-out output changes, then there exists no state-defining partition containing that junction-in output change and its junction-out output changes in a common block. If a junction-in output change is compatible with all of its junction-out output changes, then it is in a common block of the state defining partition either with all of its junction-out output changes or with none of them. These statements can be proved [10] easily, because in the opposite cases, the secondary change could not be uniquely prescribed to the junction-in output change. For, it is not known yet during a junction-in change which junction-out will follow it.

The junction-in and junction-out output changes are very easy to point out in the $B:K$ table. For, in the column of a junction-in output change there are more than one output changes in row $(K)K$. These are the junction-out changes belonging to the junction-in change determining the column.

So, in the example shown in Fig. 16 the junction-in and junction-out changes are as follows:

Junction-in	*Junction-out*
K_3^4	K_4^3, K_4
K_2	$K_3^2/2, K_2^4$
K_2^4	K_4^3, K_4
K_3^1	K_1^2, K_1^3

Let it be examined, whether the state-defining partition

$$\pi = \{(K_1^2 K_1^3 K_2^3/1 K_2^3/2 K_1^3 K_2^4)(K_4^3 K_4 K_2)(K_3 K_3^4)\}$$

obtained in the example of Fig. 15 corresponds to the conditions of junction-in and junction-out changes.

The second block contains K_4^3 and K_4; in the first block, $K_2^3/2$, K_2^4 and K_1^2, K_1^3 are together.

In consequence, to calculate an optimal state-defining partition, an optimal closed cover is to be determined starting from the maximal compatibility classes of the prescribed output changes. This cover is to be made disjoint whilst remaining closed minimizing the number of state transitions and avoiding the conflict with the above conditions of junction-in and junction-out changes. During this procedure, many variations may occur in making a cover disjoint by leaving out output changes from a compatibility class. For an advantageous selection strategy, let the K_n^m *efficiency* $h_i(K_n^m)$ of a compatibility class R_i be introduced according to the following definition: $h_i(K_n^m)$ is the number of the prescribed changes in the longest fragment of prescribed sequences which can be composed of the output changes—including K_n^m—contained by R_i.

It is obvious that

$$0 \leq h_i(K_n^m) \leq | R_i |,$$

where $| R_i |$ denotes the number of elements in R_i.

If R_i does not contain K_n^m, then

$$h_i(K_n^m) = 0$$

The meaning of $h_i(K_n^m) = 1$ is that R_i contains K_n^m, but it does not contain any other prescribed output changes which could be composed to include K_n^m in order to construct any fragments of prescribed sequences.

If K_n^m could be removed from more than one compatibility class of a cover preserving the closure property and avoiding the conflict with the junction-in, junction-out conditions, then in order to reduce the number of the state transitions, K_n^m should be removed from those compatibility classes which have the smallest value of $h_i(K_n^m)$.

The proof of this statement [10] shows that if K_n^m were removed from compatibility classes having not the smallest $h_i(K_n^m)$ then the number of the state transitions would always be equal or greater than it would be in the case of the smallest $h_i(K_n^m)$.

For example, let a closed cover of the maximal compatibility classes belonging to Fig. 15 be examined:

$$R_1 = (K_1^2 K_1^3 K_2^3/1 K_2^3/2 K_1^3 K_2^4)$$

$$R_2 = (K_1^3 K_2^3/1 K_3 K_3^4 K_4^3)$$

$$R_3 = (K_4^3 K_4 K_4^2)$$

$h_1(K_2^3/1) = 4$ because of the fragment $K_2^3/2 K_3^1 K_1^2 K_2^3/1$,

$h_2(K_2^3/1) = 3$ because of the fragment $K_2^3/1 K_3^4 K_4^3$.

So, $K_2^3/1$ is to be remowed from R_2. This removal preserves the closure property of the cover and results in the state-defining partition mentioned earlier.

Unfortunately, it may occur that the $B : K$ graph or set contains junction-out output changes which belong to a common junction-in change and are incompatible. In such cases there is a conflict with the condition of junction-in and junction-out output changes and so, it is impossible to generate a state-

K	12	23	25	32	51	43	31	24
$B(K)$	12	24 34	23	43	31	54	51	35
$(K)K$	23 25	32	51	24 23	12	31	12	43
$K(K)$	31 51	12 32	12	23	25	24	43	32
$(K)B$	24 23	43	13	34 35	12	41	12	54

Fig. 19

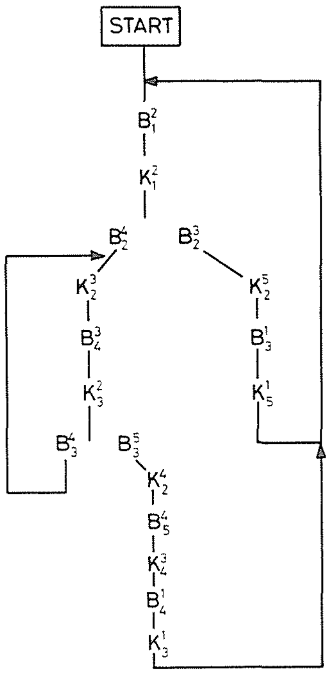


Fig. 20

defining partition for such a form of the $B^{\cdot}: K$ graph or set. In order to eliminate this difficulty, the $B:K$ graph and set are to be formally modified by introducing recurrent output changes.

For example, a $B : K$ table and the corresponding $B : K$ graph are given in Fig. 19 and 20. The junction-out changes K_2^3 and K_2^5 are incompatible if $K_2^3 \parallel K_2^5$ is not true, because

$$B(K_2^3)^{-1} : d \bigcap B(K_2^5) : d = \{X^3\} \neq \emptyset .$$

K	12	23/1	23/2	25	32	51	43	31	24
$B(K)$	12	24	34	23	43	31	54	51	35
$(K)K$	23/1 25	32	32	51	24/2	12	23/2	12	43
$K(K)$	31 51	12	32	12	23/1 23/2	25	24	43	32
$(K)B$	24 23	43	43	31	34 35	12	41	12	54

Fig. 21

Let $K_2^3/1$ and $K_2^3/2$ be introduced instead of K_2^3 according to $Q_1=\{B_2^4\}$ and $Q_2=\{B_3^4\}$ of $B(K_2^3)$.

The modified $B:K$ set and graph is shown in Fig. 21 and 22, where $K_2^3/1$ and K_2^5 are already compatible.

Let an unambiguous $B:K$ set or graph be called a *canonical* one if it is open and all of its junction-out changes belonging to common junction-in changes are compatible for every possible pair.

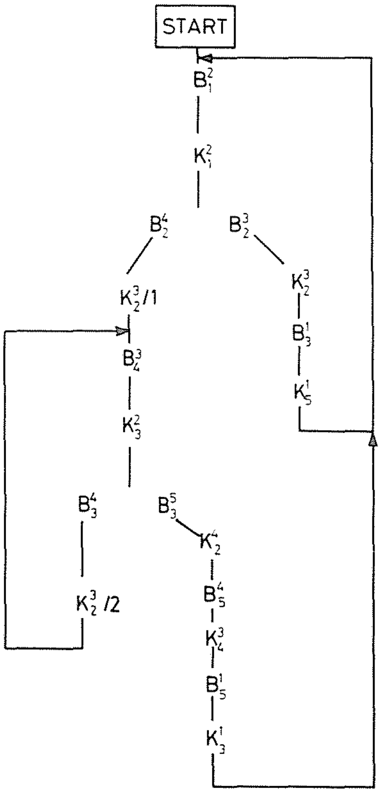


Fig. 22

The procedure for obtaining the canonical version of a $B:K$ set is based on the step-by-step selection of the output changes which are to be made recurrent.

The speed of convergence of the procedure [10] is influenced by the heuristic selection from the output changes which are to be replaced by their recurrent copies in each step.

Having obtained the canonical $B:K$ set, the method for calculating the state defining partitions provides the secondary changes $A(K_n^m)$ for every prescribed output change.

K	12	13	23/1	3	34	43	4	42	2	23/2	24	31
$B(K)$	12	13	24	36	45 [65]	54	57	71	16	67	65	71
$K(K)$	31	31	12	13	23/1 3.34	34	34 24	4	42	2	2	23/2
$(K)K$	23/1	3	34	34	43 4	34	42	2	23/2 24	31	43 4	12 13
$(K)B$	24	36 [65]	45 [65]	45 57	54 [65]	45	71	16 66	67	71 57	54 13	12
$A(K)$	1	13	13	3	32	23	2	2	21	1	12	1

Fig. 23

Let $A(K)$ denote the set of all secondary changes $A(K_n^m)$. Let $B:K:A$ denote all of the sets $B(K)$, $K(K)$, $(K)K$, $(K)B$ and $A(K)$.

For example, the $B:K:A$ set belonging to Fig. 18 is shown in Fig. 23 as a so-called $B:K:A$ *table* including also the fictive secondary changes.

A $B:K$ graph with the secondary changes can be considered as a $B:K:A$ *graph*. So, Fig. 18 represents a $B:K:A$ graph.

A $B:K:A$ graph can be called an optimal one if the number of states and state-transitions cannot be reduced whithout changing the corresponding canonical $B:K$ graph.

The method outlined in this paper for calculating the state-defining partitions has the same quality of optimality and straightforwardness as the well-known procedure [1] for minimising the number of states of incompletely specified sequential circuits.

Later in this paper, it is supposed that the $B:K:A$ set or graph is given as a specification for the realisation of the control unit. The further steps are determined by the hardware structure and by the choice between a synchronous or an asynchronous mode of operation.

As an example, the realisation steps will be summarised in the next chapter for the case of a synchronous phase register structure shown in Fig. 2.

V. Expressions for the realisation of the control unit specified by $B:K:A$ set

Let it be assumed that a prescribed output change K_n^m occurs in the state y^k determined by a state-defining partition of the control unit.

Let $\begin{Bmatrix} J_n^m \\ T_n^m \end{Bmatrix}$ denote the set of the prescribed input combinations occurring in the state y^k and $\begin{Bmatrix}\text{causing} \\ \text{inhibiting}\end{Bmatrix}$ the output change K_n^m.

It is obvious that

$$J_n^m \cap T_n^m = \emptyset,$$

because otherwise, it could not be decided, whether K_n^m occurs in y^k or does not. This would conflict with the character of the state-defining partions which defines y^k.

Let $\begin{Bmatrix} B:y^k \\ K:y^k \end{Bmatrix}$ denote the set of all prescribed $\begin{Bmatrix}\text{input} \\ \text{output}\end{Bmatrix}$ changes occurring in the state y^k.

It can be proved [10] that the sets J_n^m and T_n^m can be expressed as follows:

1) If no real (fictive) secondary change belongs to K_n^m, then

$$J_n^m = B(K_n^m):d$$

$$T_n^m = \frac{B:y^k}{B(K_n^m) \cup H(K_n^m)}:d \cup \frac{(B:y^k)^{-1}:d}{B(K_n^m):d} \tag{1}$$

2) If a real (not fictive) secondary change belongs to K_n^m, then

$$J_n^m = B(K_n^m):d$$

$$T_n^m = \frac{B:y^k}{B(K_n^m) \cup E(K_n^m)}:d \cup \frac{(B:y^k)^{-1}:d}{B(K_n^m):d} \tag{2}$$

3) If the set $B(K_n^m)$ contains at least one prescribed input section change, then

$$J_n^m = B(K_n^m):d$$

$$T_n^m = \frac{L}{B(K_n^m):d} \tag{3}$$

4) If no real (fictive) secondary change belongs to a fictive prescribed output change K_n, then

$$T_n = \emptyset. \tag{4}$$

In the above expressions, the broken line indicates that the elements of the sets in the denominators must be left out of the sets in the numerator; $E(K_n^m)$ denotes the union of all $B(K_p^r)$ sets for every K_p^r which is parallel to K_n^m and occurs also in y^k with the same secondary changes as K_n^m; $H(K_n^m)$ denotes the union of all $B(K_p^r)$ sets for every K_p^r which is parallel to K_n^m and occurs also in y^k; L denotes the set of all prescribed input combinations.

If the prescribed output change K_n^m occurs in the state y^k determined by a state-defining partition, then an *identifying Boole function* $F_n^m(X)$ can be formed as follows:

$$\begin{aligned} &F_n^m(X)=1\,, \quad \text{if} \quad X\in J_n^m \\ &F_n^m(X)=0\,, \quad \text{if} \quad X\in T_n^m \\ &F_n^m(X)=-(\text{don't care})\,, \quad \text{if} \quad X\notin J_n^m \quad \text{and} \quad X\notin T_n^m \end{aligned} \tag{5}$$

Having formed the identifying functions for every prescribed output change, Boolean expressions can be constructed for the changing conditions of every output and secondary variable [10]:

$$Z_i:1=\sum_{k=1}^{p}\,[m(y^k)\cdot S(F_{Zi}:y^k)]$$

$$Z_i:0=\sum_{k=1}^{p}\,[(m(y^k)\cdot S(F_{\bar{Z}i}:y^k)]$$

$$Y_i:1=\sum_{k=1}^{p}\,[m(y^k)\cdot S(F_{Yi}:y^k)]$$

$$Y_i:0=\sum_{k=1}^{p}\,[m(y^k)\cdot S(F_{\bar{Y}i}:y^k)]\,,$$

where $\left\{\begin{matrix} Z_i:1 \\ Z_i:0 \\ Y_i:1 \\ Y_i:0 \end{matrix}\right\}$ denotes the value of the logical (Boolean) function which is 1 only if during a prescribed $\left\{\begin{matrix}\text{output}\\ \text{output}\\ \text{secondary}\\ \text{secondary}\end{matrix}\right\}$ change, the value of the $\left\{\begin{matrix}\text{output}\\ \text{output}\\ \text{secondary}\\ \text{secondary}\end{matrix}\right\}$ variable $\left\{\begin{matrix} Z_i \\ Z_i \\ Y_i \\ Y_i \end{matrix}\right\}$ changes $\left\{\begin{matrix}\text{from 0 to 1}\\ \text{from 1 to 0}\\ \text{from 0 to 1}\\ \text{from 1 to 0}\end{matrix}\right\}$;

$\sum_{k=1}^{p}$ denotes the logical sum of the expression between the symbols [ ] according to k;
p is the number of the secondary variables;
the symbol $\cdot$ denotes the logical product (AND operation);
$m(y^k)$ represents the logical product of all secondary variables $(y_1 \ldots y_i \ldots y_p)$ containing $\begin{Bmatrix} y_i \\ \bar{y}_i \end{Bmatrix}$ if the value of y_i is $\begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$ in the secondary combination y^k;
$\begin{Bmatrix} F_{Z_i}: y^k \\ F_{\bar{Z}_i}: y^k \\ F_{Y_i}: y^k \\ F_{\bar{Y}_i}: y^k \end{Bmatrix}$ denotes the set of identifying functions belonging to such prescribed $\begin{Bmatrix} \text{output} \\ \text{output} \\ \text{secondary} \\ \text{secondary} \end{Bmatrix}$ changes which occur in the state y^k and cause the change of the value

$$\begin{Bmatrix} \text{from 0 to 1} \\ \text{from 1 to 0} \\ \text{from 0 to 1} \\ \text{from 1 to 0} \end{Bmatrix} \text{ of the } \begin{Bmatrix} \text{output} \\ \text{output} \\ \text{secondary} \\ \text{secondary} \end{Bmatrix} \text{ variable } \begin{Bmatrix} Z_i \\ Z_i \\ Y_i \\ \bar{Y}_i \end{Bmatrix};$$

$S(\quad)$ denotes the logical function which is obtained by forming the logical sum (OR operation) of all functions contained by the set in the brackets.

In the above expressions, the definition of the identifying functions is formally extended for the prescribed secondary changes. This interpreting is trivial, because every secondary change connected with an output change even if it is a fictive one.

In the synchronous phase register structure outlined in Fig. 2 the secondary combinations are restricted being in 1-from-n code. For that reason, $m(y^k)$ can be replaced by the secondary variable y_k assuming that $y_k = 1$ in the state y^k. In consequence of this:

$$Y_i : 0 = \sum_{j \neq i} Y_j : 1,$$

where $\sum_{j \neq i}$ denotes the logical sum (OR operation) for every i differring from j.

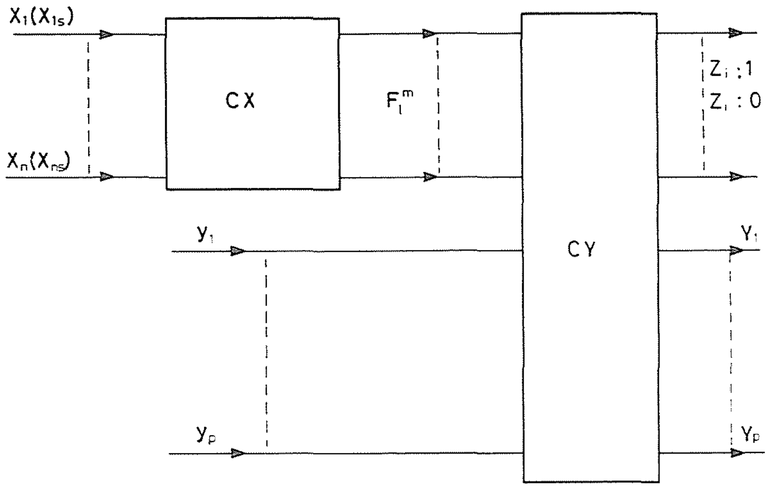


Fig. 24

So, the expressions $Y_i:0$ can be neglected and the notation Y_i can be used instead of $Y_i:1$:

$$z_i:1=\sum_{k=1}^{p}[S(F_{Z_i}:y^k)\cdot y_k]$$

$$z_i:0=\sum_{k=1}^{p}[S(F_{\bar{Z}_i}:y^k)\cdot y_k] \qquad (6)$$

$$Y_i=\sum_{k=1}^{p}[S(F_{Yi}:y^k)\cdot y_k]$$

The character of the above expressions involves a decomposition of the combinational part C in Fig. 2 as shown in Fig. 24.

Every identifying function F_n^m can be considered as representing a partial dichotomy D_n^m [10] of prescribed input combinations as follows:

$$D_n^m=\{(X\in J_n^m),\quad (X\in T_n^m)\},$$

where $X\in J_n^m$ and $X\in T_n^m$ denote all prescribed input combinations which are contained by J_n^m and T_n^m respectively.

Two identifying functions F_n^m and F_u^v can be replaced by a single identifying function in the expressions $Z_i:1$, $Z_i:0$, $Y_i:1$, $Y_i:0$ if and only if [10]
— the prescribed output changes K_n^m and K_u^v occur in different states and
— the ordered partial dichotomies D_n^m and D_u^v are compatible [4].

It also can be proved [10] that there is a compatibility relationship between F_n^m and F_u^v if they can be replaced by a single identifying function. The optimal cover, calculated from the maximal compatibility classes of the identifying functions, represents the fewest identifying functions for the realisation of the expressions $Z_i:1$, $Z_i:0$, $Y_i:1$, $Y_i:0$. It is obvious that the realisation, based on this optimal cover of the identifying functions may reduce the number of outputs from the network CX in Fig. 24. This is especially advantageous if the combinational part is constructed from PLA or memory units. The decomposition shown in Fig. 24 may have another advantage in uniform realisation. For, the network CX is independent of the secondary variables and CY accomplishes simple two-level sum-of-products operations.

For example, let the identifying functions be calculated for the $B:K:A$ set in Fig. 19 assuming the output combinations shown in Fig. 17.

The sets J_n^m and T_n^m can be determined with the help of expressions (1), (2), (3):

$$\begin{array}{rl}
J_1^2=\{X^2\} & T_1^2=\{X^3, X^6, X^4, X^1, X^7, X^5\} \\
J_1^3=\{X^3\} & T_1^3=\{X^2, X^6, X^4, X^1, X^7, X^5\} \\
J_2^3/1=\{X^4\} & T_2^3/1=\{X^2, X^3, X^6, X^1, X^7, X^5\} \\
J_3^4=\{X^5\} & T_3^4=\{X^1, X^2, X^3, X^4, X^6, X^7\} \\
J_4^3=\{X^4\} & T_4^3=\{X^7, X^1, X^6, X^5\} \\
J_4^2=\{X^1\} & T_4^2=\{X^7, X^6, X^5, X^4\} \\
J_2=\{X^6\} & T_2=\{X^4, X^5, X^7, X^1\} \\
J_2^3/2=\{X^7\} & T_2^3/2=\{X^1, X^5, X^6, X^2, X^3, X^4\} \\
J_3^1=\{X^1\} & T_3^1=\{X^7, X^5, X^6, X^2, X^3, X^4\} \\
J_2^4=\{X^5\} & T_1^4=\{X^7, X^1, X^6, X^2, X^3, X^4\}
\end{array}$$

Among the fictive output changes, there is only one (K_2) which must be taken into consideration, because the others are without secondary changes and so, they have no effect.

Thus, the identifying functions are determined according to the definition (5) and the functions $S(F_{Zi}:y^k)$, $S(F_{\bar{Z}_i}:y^k)$, $S(F_{Yi}:y^k)$ can be obtained:

$$S(F_{Z1}:y^1)=F_2^4; \quad S(F_{Z1}:y^3)=F_3^4;$$

$$S(F_{\bar{Z}1}:y^2)=F_4^3+F_4^2$$

$$S(F_{Z2}:y^1)=F_3^1$$

$$SF_{\bar{Z}2}:y^1)=F_1^2+F_1^3$$

$$S(F_{Z3}:y^1)=F_1^3+F_2^3/1+F_2^3/2; \quad S(F_{Z3}:y^2)=F_4^3$$

$$S(F_{\bar{Z}3}:y^1)=F_3^1; \quad S(F_{Z3}:y^3)=F_3^4$$

$$S(F_{Z4}:y^1)=F_1^2; \quad S(F_{Z4}:y^3)=F_3^4$$

$$S(F_{\bar{Z}4}:y^1)=F_2^3/1+F_2^3/2;\quad S(F_{Z4}:y^2)=F_4^3$$
$$S(F_{Y1}:y^2)=F_2$$
$$S(F_{Y2}:y^1)=F_2^4;\quad S(F_{Y2}:y^3)=F_3^4$$
$$S(F_{Y3}:y^1)=F_2^3/1+F_1^3;\quad S(F_{Y2}:y^3)=F_4^3$$

Substituting into the expressions (6):

$$\begin{gathered}
Z_1:1=F_2^4y_1+F_3^4y_3;\quad Z_1:0=(F_4^3+F_4^2)y_2\\
Z_2:1=F_3^1y_1;\quad Z_2:0=(F_1^2+F_1^3)y_1\\
Z_3:1=(F_1^3+F_2^3/2)y_1+F_4^3y_2;\quad Z_3:0=F_3^1y_1+F_3^4y_3\\
Z_4:1=F_1^2y_1+F_3^4y_3;\quad Z_4:0=(F_2^3/1+F_2^3/2)y_1+F_4^3y_2\\
Y_1=F_2y_2;\quad Y_2=F_2^4y_1+F_3^4y_3;\quad Y_3=(F_2^3/1+F_1^3)y_1+F_4^3y_2
\end{gathered}\tag{7}$$

Without detailing the steps of compatibility checking and covering algorythm, the disjoint blocks of the optimal cover of the identifying functions are as follows:

$$\begin{array}{ll}
R_1:(F_1^2) & R_4:(F_3^4F_2^4)\\
R_2:(F_1^3) & R_5:(F_3^1F_4^2)\\
R_3:(F_2^3/1F_4^3) & R_6:(F_2^3/2)\\
 & R_7:(\bar{F}_2)
\end{array}$$

With the help of a simple algorythm [10], a set of identifying functions $(F^1, F^2 \ldots F^7)$ can be constructed representing the original identifying functions contained in the blocks $R_1, R_2 \ldots R_7$ respectively. For example, F^5 represents F_3^1 and F_4^2. So, F_3^1 and F_4^2 can be replaced by F^5 in the expressions (7).

Executing all of the possible substitutions *b* F^1, $F^2 \ldots F^7$ in the expressions (7):

$$\begin{gathered}
Z_1:1=F^4(y_1+y_3);\quad Z_1:0=(F^3+F^5)y_2\\
Z_2:1=F^5y_1;\quad Z_2:0=(F^1+F^2)y_1\\
Z_3:1=(F^2+F^3+F^6)y_1+F^3y_2;\quad Z_3:0=F^5y_1+F^4y_3\\
Z_4:1=F^1y_1+F^4y_3;\quad Z_4:0=(F^3+F^6)y_1+F^3y_2\\
Y_1=\bar{F}^7y_2,\quad Y_2=F^4(y_1+y_3),\quad Y_3=(F^3+F^2)y_1+F^3y_2
\end{gathered}\tag{8}$$

Applying the identifying functions $F^1, F^2, \ldots F^7$ instead of the original ones, the number of outputs of the network CX (Fig. 24) can be reduced from 10 to 7.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X^1	0	0	1	0	0	0	0
X^2	1	0	1	0	0	1	0
X^3	0	0	0	0	0	0	1
X^4	1	0	0	0	1	0	0
X^5	1	1	1	0	1	1	0
X^6	1	1	1	1	1	0	1
X^7	1	0	1	1	1	1	1

Fig. 25

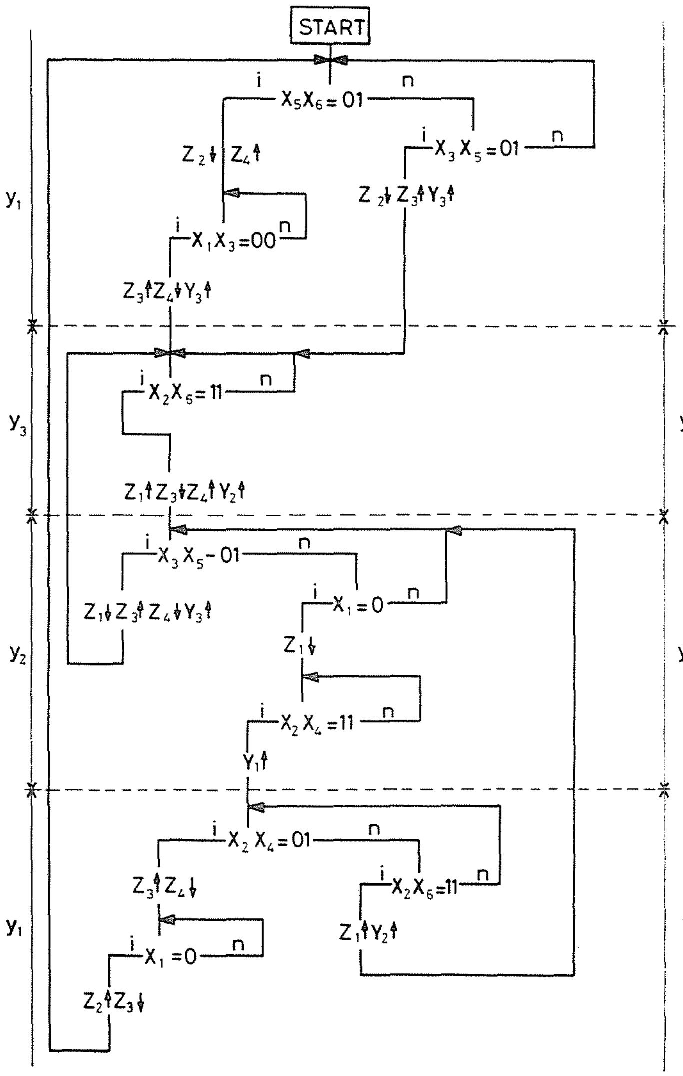


Fig. 26

The realisation of the identifying functions $F^1, F^2, \ldots F^7$ can be accomplished easily, because the sets J^i and T^i are given according to the definition (5). Assuming the prescribed input combinations shown in Fig. 25, one of the possible methods [9] yields the solution as follows:

$$F^1 = \bar{X}_5 X_6$$
$$F^2 = \bar{X}_1 \bar{X}_3$$
$$F^3 = \bar{X}_3 X_5$$
$$F^4 = X_2 X_6$$
$$F^5 = \bar{X}_1$$
$$F^6 = \bar{X}_2 X_4$$
$$F^7 = X_2 X_4$$

Introducing the above solution into the expressions (8)

$$Z_1: 1 = X_2 X_6 (y_1 + y_3); \quad Z_1: 0 = (\bar{X}_3 X_5 + \bar{X}_1) y_2$$
$$Z_2: 1 = \bar{X}_1 y_1; \quad Z_2: 0 = (\bar{X}_5 X_6 + \bar{X}_3 X_5) y_1$$
$$Z_3: 1 = (\bar{X}_1 \bar{X}_3 + \bar{X}_3 X_5 + \bar{X}_2 X_4) y_1 + \bar{X}_3 X_5 y_2$$
$$Z_3: 0 = \bar{X}_1 y_1 + X_2 X_6 y_3$$
$$Z_4: 1 = \bar{X}_5 X_6 y_1 + X_2 X_6 y_3;$$
$$Z_4: 0 = (\bar{X}_3 X_5 + \bar{X}_2 X_4) y_1 + \bar{X}_3 X_5 y_2$$
$$Y_1 = X_2 X_4 y_2; \quad Y_2 = X_2 X_6 (y_1 + y_3);$$
$$Y_3 = (\bar{X}_3 X_5 + \bar{X}_1 \bar{X}_3) y_1 + \bar{X}_3 X_5 y_2$$

The flow chart in Fig. 26 can be considered as a graphical illustration of the above expressions, but it does not specify uniquely the control procedure unless we know the prescribed sequences of the input and output changes.

VI. Special features of the method

VI.1. Applying an output decoder

If the hardware structure stores the output combinations, then an output decoder placed after the output flip-flops may reduce the number of blocks in the optimal state-defining partitions. The reason for this is that the storing effect of the output flip-flops can contribute to the representation of the states and so, less secondary combination may be required to distinguish the states.

Based on the algorythm PARKOMP, we can say, which incompatibility situations between K_n^m and K_r^r would turn into compatibility, if the relationships $K_n^m \| K_p^r$, $K_n^m \| K_p K_p^r \| K_n$ were valid. These extra conditions for the output codes are to be fulfilled at the input of the output decoder. Thus, the specification of the output decoder should ensure the optimal output combinations (at the input of the output decoder) to be checked by PARKOMP. Obviously, conflicts may arise, since these conditions are to be fulfilled simultaneously. Thus, some trial-and-error steps may be unavoidable. However, the procedure for calculating the optimal state-defining partition [10] makes these steps easy to execute. For example, let $K_n^m \| K_p^r$ and $K_p^r \| K_s^t$ be true separately. In this case, $K_n^m \| K_p^r$ and $K_p^r \| K_s^t$ cannot be ensured simultaneously by any assign-variations of output combinations if and only if $m=s$ and $n=t$ [10].

VI.2. Handling of incompletely specified input and output changes

A prescribed combination is incompletely specified if it has at least one bit which is not specified (don't care). Thus, an incompletely specified combination represents several completely specified combinations. It is obvious that using incompletely specified combinations, a more concise initial description can be obtained for a control procedure, because the $B:K$ set formally consists of less elements.

If the state-defining partition is determined applying incompletely specified input changes, then during the calculation the sets J_n^m and T_n^m may become not disjoint. For example, if

	X_1	X_2	X_3	X_4	X_5
X^j:	1	0	–	1	1
X^h:	1	–	1	–	–

and

$$X^j \in J_n^m \quad X^h \in T_n^m,$$

then the input combination 10111 would be contained by both J_n^m and T_n^m which is a contradiction. In this case, the incompletely specified input changes X^j and X^h are said to be unseparated [10]. In such cases, the state-defining partition can be made suitable for the realisation in a heuristic way by removing states from the blocks, where the not separated pairs occur, and forming new blocks.

This heuristic step is avoidable if even during the execution of the algorythm PARKOMP, the unseparated input combinations are taken into consideration.

In this case, the output combinations K_n^m and K_p^r are always considered to be incompatible if the sets $B(K_n^m):d$ and $B(K_p^r)^{-1}:d$ or $B(K_n^m)^{-1}:d$ and $B(K_p^r):d$ are unseparated. In this way, the optimality of the state-defining partition is impossible to ensure, but the concise description may compensate for this disadvantage.

The incompletely specified output changes do not cause any contradictions during the design procedure and the unspecified bits are fixed at the end in the most economic way. Besides the concise description, another advantage is that the incompletely specified output changes may simplify the functions $S(F_{Zi}:y^k)$ and $S(F_{\bar{Z}_i}:y^k)$. For example, the identifying function of an incompletely specified output change K_p^r can be left out from the sums $S(F_{Zi}:y^k)$ and $S(F_{\bar{Z}_i}:y^k)$ if Z_i is unspecified in Z^r [10].

VI.3. Handling of the prescribed input and output sequences given in separate fragments

In the $B:K$ table or set, the continuity of the changes can be represented — if not otherwise defined — by the indices of the adjacent changes. If the initial definition of the control procedure is given by separate fragments of the prescribed changes, then they can be joined together by defining extra input section changes between the separate fragments. In this case the continuity of the output changes also must be ensured by defining additional output changes between the fragments.

Obviously, these extra prescriptions affect only the changes which are not mentioned in the initial definition of the control procedure. Therefore, in this way, a kind of fixing of the "don't care" situations is carried out.

The rules for the calculation of the $B:K$ table provide a systematic method for determining the changes which are not excluded in the initially empty cells of the $B:K$ table. The selection from the possible solutions is to be performed by checking the resulting specifications in order to fix the fewest initially unspecified situations.

For example, it is advantageous if the initially empty cells of the $B:K$ table are filled with at most one extra change. The simplest resulting specification will be obtained if the least and the shortest prescribed sequences are formed, starting and finishing with the initial input combinations. In this way, the specification for the synthesis procedure may also become more rigorous than the original one, but it is not necessary to form a coherent specification by intuition.

VII. Examples for illustration

At the Department of Process Control of TU Budapest, a design program-package has been developed for testing the method outlined in this paper. The self-contained parts of the program-package can be joined together in an interactive way. The main parts execute the following design steps:

— the construction of the canonical $B:K$ set and graph on the basis of the fragments of prescribed input and output changes,

— the algorythm PARKOMP and finding an optimal cover for the prescribed output changes,

— the calculation of an optimal state-defining partition,

— formulating the logic expressions for uniform hardware structures.

The main steps of the method are summarised by the flow chart in Fig. 27.

Example 1

As an example of the method and the usage of the program package, let the control unit of the system shown in Fig. 28 be considered. The prescribed function of the control unit is as follows:

— The information $INF1$ and $INF2$ from the devices $P1$ and $P2$ is transferred to the output INF depending on the value of the selection signal V. If $V=1$ then $INF1$ will be loaded into the register by the value $RB=1$.

The signals ST_i and K_i from the devices $P1$ and $P2$ respectively are handshaking-pairs. $K_i=1$ is acknowledged by $ST_i=1$ only then, when the information is already stable on the lines INF_i.

— The control unit starts the devices $P1$ and $P2$ by the signals $ST1$ and $ST2$, respectively depending on the value of the signal PV. ($PV=1$ assigns $P1$).

The starting is initialised by a demand from the environment represented by the value $KR=1$. The handshaking-pair of KR is the signal KRV acknowledging the demand of the environment of the system.

The control unit produces $K=1$ for the environment if the information from the device, assigned by PV, is already stable on the lines INF. The values K = and PV must not change until a new $KR=1$ occurs:

— If $KR=1$ occurs while the control unit is still processing the effect of an earlier $KR=1$ received for the opposite device, then the control unit must respond to the environment by the busy signal $F=1$ instead of $KRV=1$. In this case the processing of the earlier $KR=1$ must be completed without any disturbance, but subsequently, the value $KRV=1$ must be transmitted to the environment. This $KRV=1$ represents the message to the environment that the selection, rejected earlier by the busy signal, is now possible.

— If $KR=1$ occurs while the control unit is still processing the effect of an earlier $KR=1$ received for the same device, then the processing of the earlier

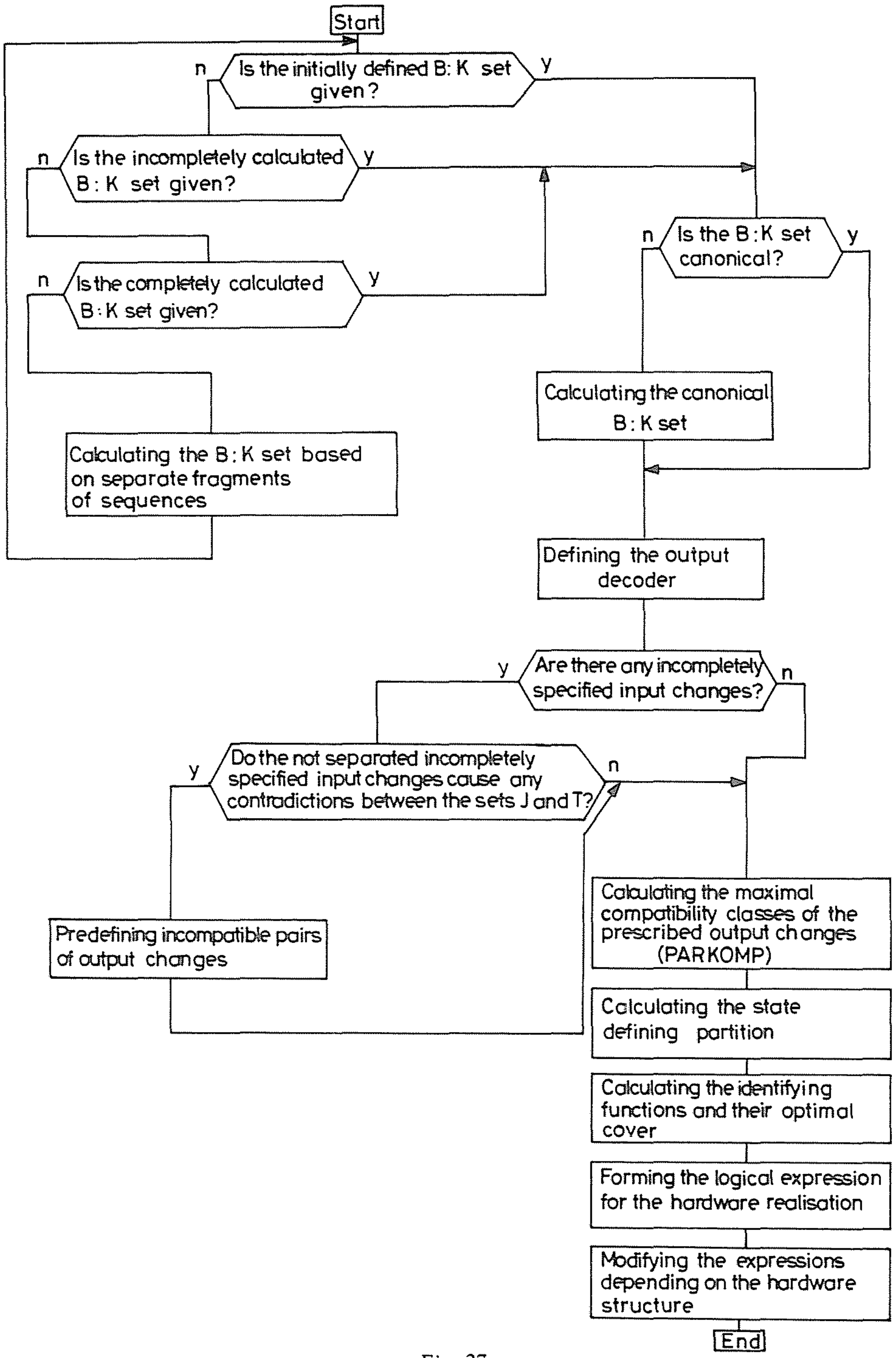

Start
Is the initially defined B: K set given?
y
n
Is the incompletely calculated B: K set given?
y
n
Is the completely calculated B: K set given?
y
n
Calculating the B: K set based on separate fragments of sequences
Is the B: K set canonical?
n
y
Calculating the canonical B: K set
Defining the output decoder
Are there any incompletely specified input changes?
y
n
Do the not separated incompletely specified input changes cause any contradictions between the sets J and T?
y
n
Predefining incompatible pairs of output changes
Calculating the maximal compatibility classes of the prescribed output changes (PARKOMP)
Calculating the state defining partition
Calculating the identifying functions and their optimal cover
Forming the logical expression for the hardware realisation
Modifying the expressions depending on the hardware structure
End

Fig. 27

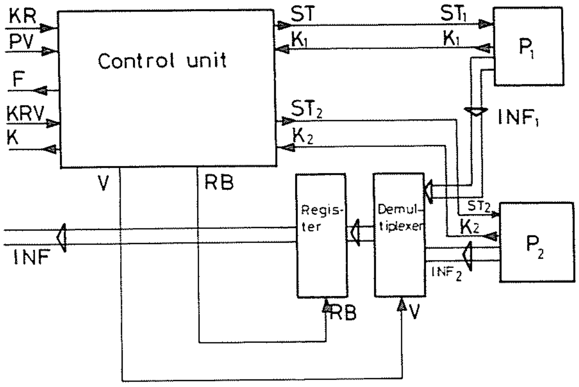


Fig. 28

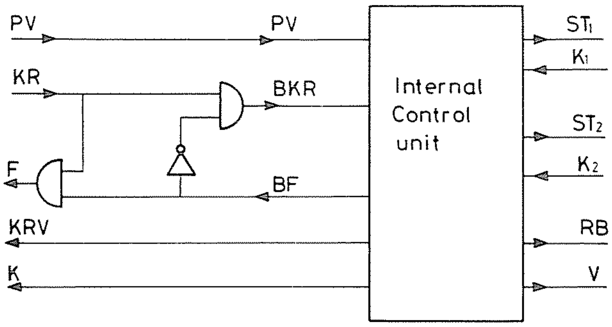


Fig. 29

$KR=1$ must be completed without any disturbance, but instead of producing $K=1$ the control unit must perform a new start for the same device. After having processed the new start, the control unit must send $K=1$ as usual unless during the processing, there occurs no $KR=1$ for the same device.

— The above specification prescribes the following handshaking pairs:

$$ST1-K1$$

$$ST2-K2$$

$$KR-KRV, F(K)$$

The signal RB is supposed to be a pulse. $K=1$ is acknowledged by $KR=1$, but K has no effect to the further changes of KR. So, the handshaking relation between K and KR is not mutual.

The control procedure can be simplified by introducing the internal control unit with extra gates shown in Fig. 29.

Signals / Combinations	X_1	X_2	X_3	X_4
	PV	BKR	K_1	K_2
X^1	–	0	–	–
X^2	–	1	–	–
X^3	1	0	–	–
X^4	0	0	–	–
X^5	1	0	1	0
X^6	0	0	0	1

Signals / Combinations	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
	BF	KRV	K	$ST1$	$ST2$	RB	V
Z^1	0	0	–	0	0	0	–
Z^2	1	1	0	0	0	0	–
Z^3	1	0	0	0	0	0	–
Z^4	1	0	0	1	0	0	1
Z^5	1	0	0	0	1	0	0
Z^6	1	0	0	1	0	1	1
Z^7	1	0	0	0	1	1	0
Z^8	1	0	1	0	0	0	–

Fig. 30

In the initial state the values $BKR=K1=K2=K0$ and $BF=KRV=ST1=ST2=RB=0$ are assumed. In the initial state the signals PV and V may have "don't care" values.

Defining the prescribed input and output combinations according to Fig. 30, the $B:K$ graph of the control procedure is shown in Fig. 31.

The maximal compatibility classes calculated by the program are as follows:

(1,9) (1,10) (2) (3) (4) (5,6) (5,8) (6,7) (7,8)

For the sake of a simple formal description, the prescribed output changes are represented by the numbers 1 . . . 10 according to the correspondence shown in Fig. 31.

It can be seen that the program establishes incompatibility between the junction-out changes 3 and 4 related to the common junction-in output change 2. The reason for this is that B_1^3 and B_1^4 are unseparated because of the "don't care" bits. In such cases, the algorythm PARKOMP at first formally excludes the compatibility relation between the prescribed output changes affected by the unseparated input changes. The incompatibility of 9 and 10 can be explained in the same way.

However, it can be seen from the prescribed function of the control unit that the relationships $3 \sim 4, 2 \sim 4, 2 \sim 3, 9 \sim 10$ do not cause any contradictions and so, they can be allowed during the compatibility checking. The program has an interactive possibility for predefining compatibility relations before executing the algorythm PARKOMP. Doing this for the above relationships, the following new maximal compatibility classes are obtained:

$$(1, 9, 10)\,(2, 3, 4)\,(5, 6)\,(5, 8)\,(6, 7)\,(7, 8)\,,$$

where there are no further contradictions arising from the conditions of

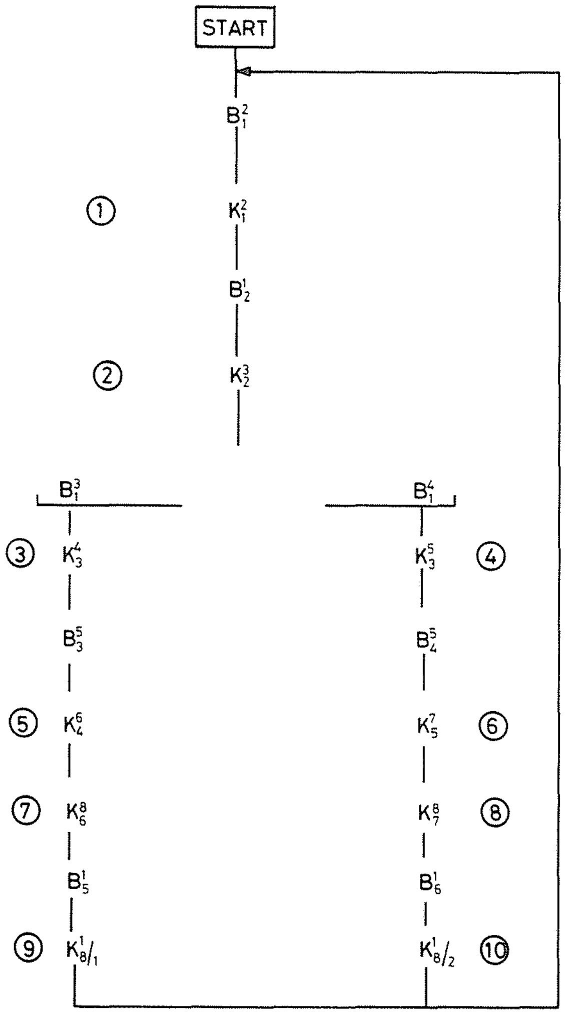


Fig. 31

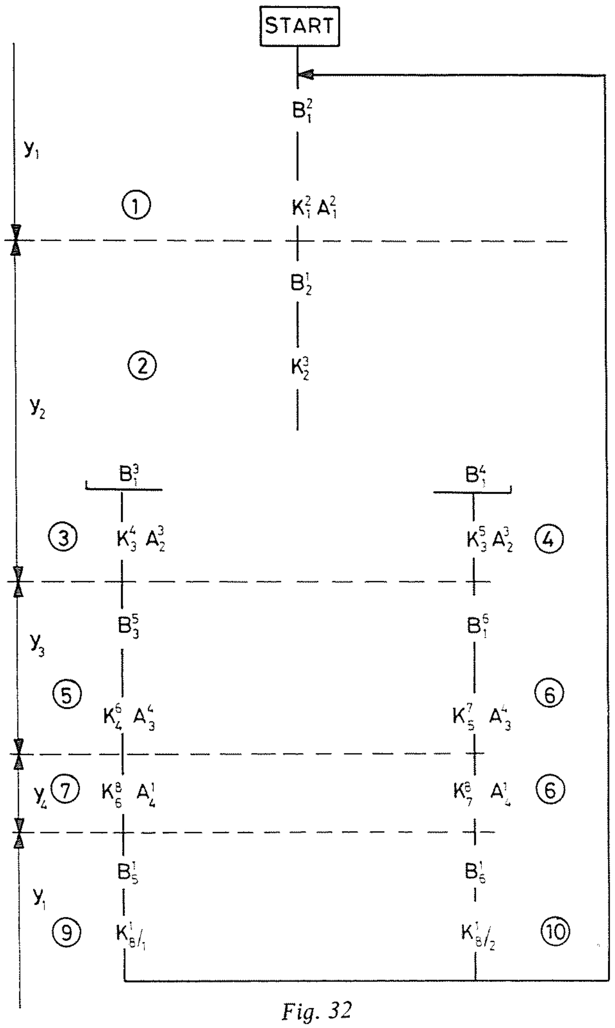


Fig. 32

junctions-in and junction-out output changes. The program provides the optimal state-defining partition:

$$(1, 9, 10)\,(2, 3, 4)\,(5, 6)\,(7, 8)$$

Assuming a synchronous phase register structure, the optimal $B : K : A$ graph is shown in Fig. 32.

Without detailing the steps for the calculation of the identifying functions, their optimal cover and the sums $S(F_{Zi} : y^k)$, $S(F_{Zi} : y^k)$, $S(F_{Yi} : y^k)$, let only the expression $Z_i : 1$, $Z_i : 0$, $Y_i : 0$ be listed:

$$BF : 1 = \overline{BKR} \cdot y_2\,; \quad BF : 0 = \overline{BKR} \cdot y_1$$

$$KRV\colon 1 = BKR \cdot y_1\,; \quad KRV\colon 0 = \overline{BKR} \cdot y_2$$

$$K : 1 = y_4\,; \quad K1 : 0 = BKR \cdot y_1$$

$$ST1 : 1 = PV \cdot \overline{BKR} \cdot y_2\,; \quad ST1 : 0 = y_4$$

$$ST2 : 1 = PV \cdot \overline{BKR} \cdot y_2\,; \quad ST2 : 0 = y_4$$

$$RB : 1 = (PV \cdot K1 \cdot \overline{K2} + \overline{PV} \cdot \overline{K1} \cdot K2) \cdot y_3\,; \quad RB : 0 = y_4$$

$$V\colon 1 = PV \cdot \overline{BKR} y_2 \quad V\colon 0 = \overline{PV} \cdot \overline{BKR} \cdot y_2$$

$$Y_1 = y_4$$

$$Y_2 = BKR \cdot y_1$$

$$Y_3 = \overline{BKR} \cdot y_2$$

$$Y_4 = (PV \cdot K1 \cdot \overline{K2} + \overline{PV} \cdot \overline{K1} \cdot K2) \cdot y_3$$

Example 2

A $B:K$ graph is given in Fig. 33. Let the state-defining partition be determined assuming the prescribed input and output combinations shown in Fig. 34.

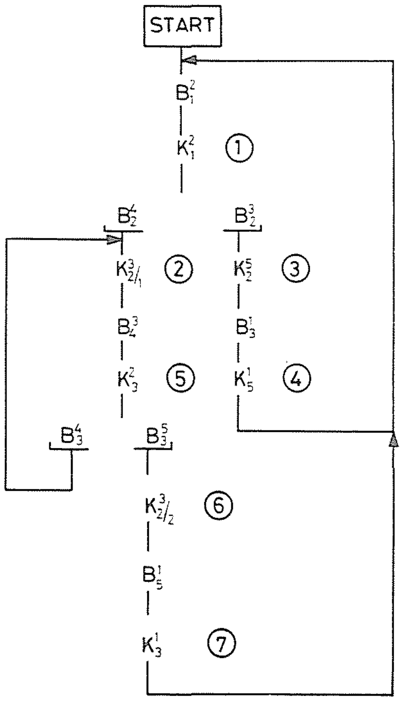


Fig. 33

Signals / Combinations	X_1	X_2	X_3
X^1	0	0	0
X^2	0	0	1
X^3	0	1	0
X^4	0	1	0

Signals / Combinations	X_1	X_2	X_3
Z^1	0	0	0
Z^2	0	0	1
Z^3	0	1	0
Z^4	0	1	1
Z^5	1	0	0

Fig. 34

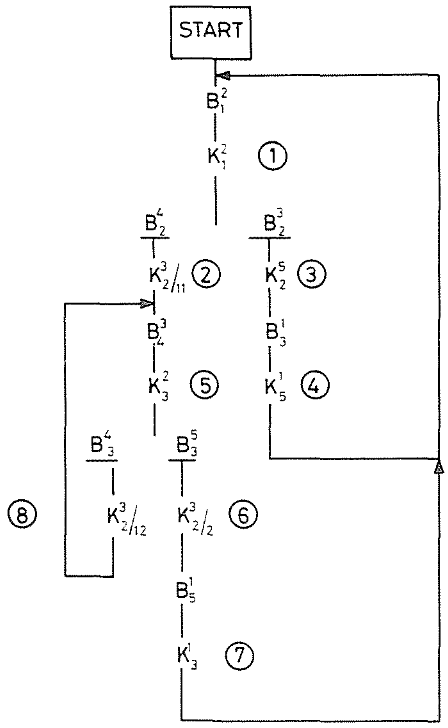


Fig. 35

The program calculates the following maximal compatibility classes:

$$(1, 2, 4, 6, 7)\,(1, 2, 5, 6, 7)\,(1, 3, 4, 7),$$

where the junction-out changes 2 and 3 related to the junction-in change 1, are not in a common compatibility class.

It means that the $B:K$ graph is not a canonical one. The problem can be avoided by introducing recurrent changes instead of $K^3_2/1$. The modified $B:K$ graph is shown in Fig. 35 which yields new maximal compatibility classes by restarting the program:

$$(1, 2, 3, 4, 7)\,(1, 2, 4, 6, 7, 8)\,(1, 4, 5, 6, 7)$$

The program provides two state defining partitions:

$$(1, 2, 3, 4)\,(6, 7, 8)\,(5)$$

and

$$(1, 2, 3, 4, 7)\,(6, 8)\,(5)$$

Example 3

A $B:K$ graph is given in Fig. 36. Let the optimal $B:K:A$ graph be constructed assuming the prescribed input and output combinations shown in Fig. 37.

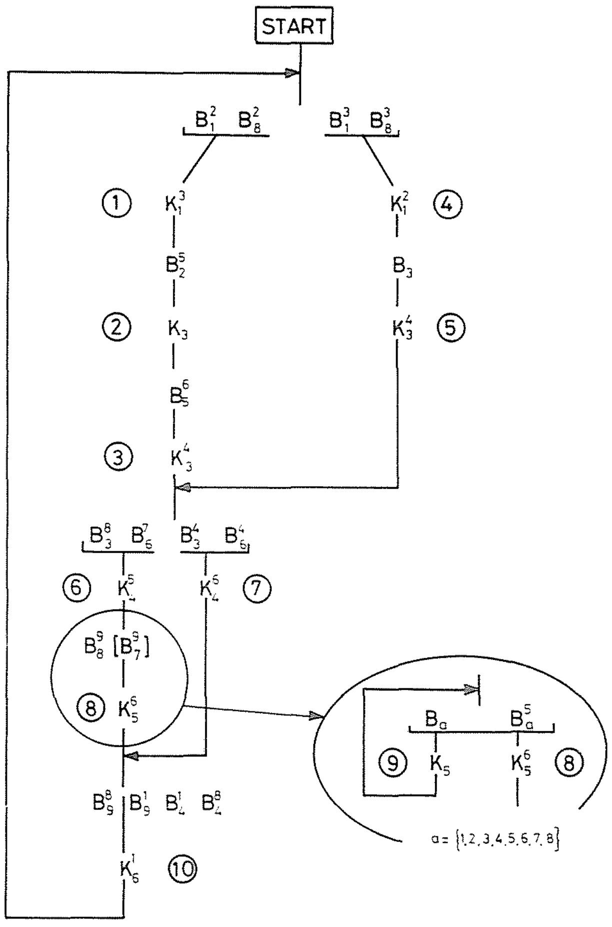


Fig. 36

Signals / Combinations	X	X	X	X
X	0	0	1	0
X^2	1	0	1	0
X^3	0	0	0	0
X^4	1	0	0	0
X^5	1	1	1	0
X^6	1	1	1	1
X^7	0	1	1	1
X^8	0	1	0	0
X^9	1	0	1	1

Signals / Combinations	Z_1	Z_2	Z_3	Z_4
Z^1	0	0	0	0
Z^2	1	1	0	0
Z^3	1	0	1	0
Z^4	1	0	0	1
Z^5	1	0	1	1
Z^6	0	0	1	1

Fig. 37

The maximal compatibility classes of the prescribed output changes are:

$$(1, 2, 3, 4, 10)\,(1, 2, 3, 5, 7, 10)\,(1, 2, 5, 8)\,(2, 3, 5, 6, 7)$$
$$(2, 5, 6, 8)\,(6, 8, 9)\,.$$

The program provides an interactive possibility for checking the closure property of a disjoint cover chosen by intuition. In this case there is a relatively large number of possible ways of forming optimal state-defining partitions. Let the solution below be tested to determine, whether it could be a state-defining partition or not:

$$(1, 2, 4, 10)\,(3, 5, 6, 7)\,(8, 9)$$

These compatibility classes do not realise some compatibility relationships contained by the maximal compatibility classes: $2\sim 3, 6\sim 8, 6\sim 9, 7\sim 10$.

The question is, whether these neglected relationships would make the cover unclosed ot not.

The program can be used for this test by restarting it after having prescribed the neglected relations as incompatibility relationships for the algorythm PARKOMP.

As a result, new maximal compatibility classes are obtained:

$$(1, 2, 4, 10)\,(1, 2, 5, 7)\,(1, 2, 5, 10)$$
$$(2, 5, 6, 7)\,(1, 3, 4, 10)\,(1, 3, 5, 7)$$
$$(1, 3, 5, 10)\,(1, 5, 8)\,(8, 9)\,(3, 5, 6, 7)\,,$$

where each compatibility relationship of the solution, chosen by intuition, is realised. Thus, the neglected compatibility relationships do not exclude the existence of the compatibility classes chosen by intuition.

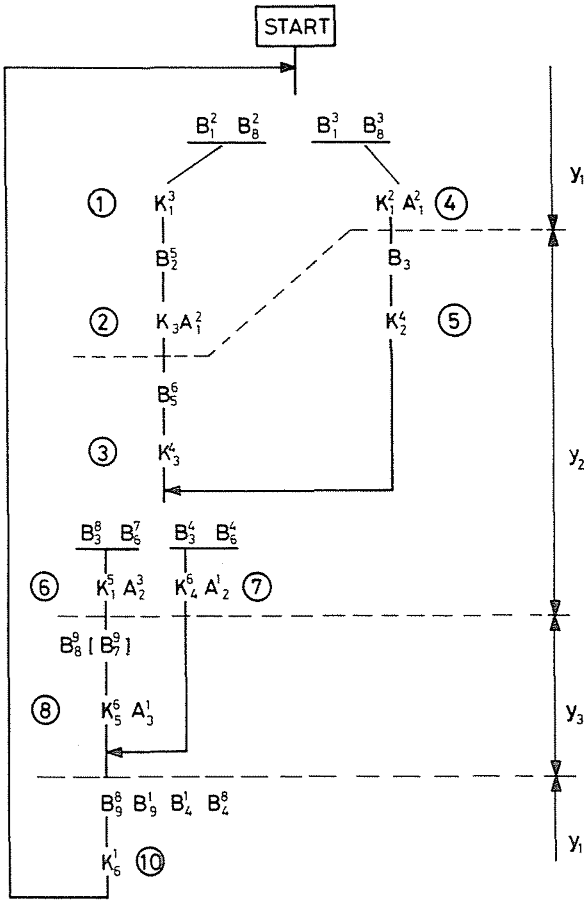


Fig. 38

Therefore the closure property is proved and the solution can be considered as a state-defining partition that yields the $B:K:A$ graph shown in Fig. 38.

References

1. McCluskey, E. J.: Introduction to the Theory of Switching Circuits, McGraw-Hill Book Company, 1965.
2. Clare, C. R.: Designing Logic Systems Using State Machines, McGraw-Hill Book Company, 1973.
3. Wendt, S.: Entwurf komplexer Schaltwerke, Springer Verlag, 1974.
4. Unger, S. H.: Asynchronous Sequential Switching Circuits, Wiley-Interscience, 1969.
5. Grass, W.: Steuerwerke (Entwurf von Schaltwerken mit Festwertspeichern), Springer-Verlag, 1978.

6. KALMÁR, P.: A Phase-State Reduction and Assignment Method Based on the Flow Chart of Control Units, Per. Pol. El. Eng. *20.*, 365–376 (1976)
7. TERPLÁN, S.: A Design Method of Asynchronous Sequential Circuits Based on Flow Diagram, MTA SZTAKI Reports 137/1982. Budapest, 1982.
8. ARATÓ, P.–TERPLÁN, S.–KALMÁR, P.–GRANTNER, J.: Methoden zum Entwurf logischer Schaltungen aus gegebenen Ablaufplänen, Zeitschrift für elektrische Informations und Energietechnik *7*, 426–436 (1977)
9. ARATÓ, P.: A Combinational Network Synthesis Method Based on Threshold Logic, Per. Pol. El. Eng. *20*, 325–353 (1976)
10. ARATÓ, P.: Control Unit Design Based Prescribed inpu: and Output Changes, Dissertation for DSc degree (in Hungarian) Budapest, 1984
11. BOCHMANN, D.–POSTHOFF, C.: Binare dynamische Systeme, Akademie-Verlag, Berlin, 1981.

Prof. Dr. Péter ARATÓ H-1521 Budapest