Automation and Remote Control, Vol. 63, No. 10, 2002, pp. 1637–1651. Translated from Avtomatika i Telemekhanika, No. 10, 2002, pp. 120–136.
Original Russian Text Copyright © 2002 by Astaf'ef, Levin, Matrosova, Sinel'nikov.

Self-Testing Automaton Networks: Their Design in Programmable Logical Matrices

M. V. Astaf'ef[*], **I. S. Levin**[**], **A. Yu. Matrosova**[*], **and V. E. Sinel'nikov**[***]

[*] *Tomsk State University, Tomsk, Russia*
[**] *Tel Aviv University, Tel Aviv, Israel,*
[***] *Technology Education Center, Holon, Israel*

Received March 16, 2000

Abstract—The design of self-testing synchronous automaton networks in a base of programmable logical matrices is studied. Self-testability of the network component is ensured by encoding the automaton internal states by equilibrium codes and elongating the input state codes. Therefore, the microprogram description of the operation of the component is transformed into a positive monotonic system of disjunctive normal forms, i.e., design specification for the component. Precisely monotonic disjunctive normal forms ensure the monotonic generation of solitary constant faults in programmable logical matrices and input poles of the automaton component at the component outputs and steady propagation of the aftereffects of a faults from its emergence point in some component up to the network outputs. Therefore, testers can be used only for external components, whose outputs are the network outputs, and only the output of these components can be observed without regard for their internal states.

1. INTRODUCTION

A self-testing discrete device usually includes a functional part for implementing certain functions and a control part (tester) for observing the outputs of the functional part. We use a synchronous discrete device with memory as the functional part and self-testing testers identifying Berger or equilibrium codes as the control part.

Network components are realized by programmable logical matrices (PLM) and D-triggers on feedback lines. Self-testing testers are also realized in a PLM base.

The design of a self-testing synchronous automaton network is reduced to the design of self-testing synchronous automaton—a network component. Solitary constant faults are admitted in programmable logical matrices, poles of D-triggers, and input and output poles of a component. Self-testability of a component is incorporated in its design.

We assume that the design specification is given in the form of a state transition graph (STG)—the analog of the microprogram description [1]. The inputs and outputs of an automaton in the state transition graph are coded. We shall encode the internal states of an automaton by equilibrium codes and "elongate" the input states of a component, preserving the same number of conjunctions in the design specification as in the STG description.

Self-testability of a component is incorporated by deriving a positive monotonic system of disjunctive normal forms (DNF) for the design specification in a PLM base. In the sequel, we refer to a system as monotonic, tacitly assuming that it is positive. A monotonic system of disjunctive normal forms can always be determined from the STG description.

We shall describe several useful properties of the components of an automaton network realizing a monotonic system. These properties are helpful in designing self-testing automaton networks with low cost on optional equipment.

(1) A component, which realizes a monotonic system in programmable logical matrices and D-triggers, monotonically develops the faults described above at its outputs. In other words, the vector β of outputs of an operative device at a certain instant t and the vector β^v of outputs (at the same instant t) in the presence of a fault v are comparable [2]: $\beta \le \beta^v$ (1-monotonic development), $\beta \ge \beta^v$ (0-monotonic development).

(2) Every fault of the class preserves its monotonicity. In other words, $\beta = \beta^v$ for the pattern α on which no fault is developed. On every test pattern α of the fault, the condition $\beta < \beta^v$ is satisfied for 1-monotonic fault development and the condition $\beta^v < \beta$ for 0-monotonic fault development (a fault cannot appear as a 1-monotonic fault on one test pattern α_1 and as a 0-monotonic fault on another test pattern α_2).

(3) Certain faults may be undetectable in the operation range of a component during observation of the outputs of the component. Their accumulation does not hide newly generated faults.

(4) A component is "transparent" to monotonic changes in input patterns. Therefore, a 1(0)-monotonic change in the input pattern of an automaton component may result in a 1(0)-monotonic change in its output pattern (not necessarily at the same instant), or has no influence on the outputs of the automaton component. "Transparency" of a component is preserved in the presence of undetectable faults in the component. This property is responsible for the monotonic propagation of the aftereffects of a fault from the place of fault development to the circuit outputs.

These properties of a component are helpful in designing self-testing automaton networks, using testers only at external components, whose outputs are the network outputs. Moreover, there is no need to observe the internal states of these components; it suffices to connect a tester to the component outputs, as described in [3] for special automata, i.e., automata that are defined by a moderate-power output alphabet.

The set of input poles of an automaton network widens if the codes of input states are "elongated." It is not possible to avoid the extension of the space of input variables (the set of input poles) in an arbitrary STG description [4]. At least one supplementary input variable is necessary [5]. The use of even (odd) codes [5] for the input states of an automaton considerably complicates the DNF system—design specification of the component. In this case, the DNF system contains conjunctions that cannot be joined by input variables, but contain all input variables.

Unlike in [5], we increase the number of supplementary input variables, retaining, as a rule, the same crystal area that is required for realizing the STG description in a PLM base without self-testability of the component. Here we mean the internal component, whose outputs are not the outputs of the automaton network. The crystal area can be increased, if necessary, in realizing self-testing external components (a) by transforming the output vectors of a component into equilibrium or Berger codes and (b) by the use of testers for observing the outputs of external components.

The approach of [6] to designing self-testing automaton networks requires a tester in every component of the network and observation of the state of the outputs and feedback lines of the component. But this approach reduces the number of supplementary input variables of a components compared to our method. The number of supplementary input variables can be reduced if STG description is used to derive a DNF system such that conjunctions with different characteristics cannot be joined and the system is not monotonic. Reducing the number of input variables, we increase the cost of optional self-testing equipment in the network.

We assume that every automaton network can be transformed into a self-testing network. The transformation consists of replacing the STG description of every component by a monotonic system of normal disjunctive forms and subsequent realization of the system in a PLM base, and joining testers to external component outputs that are the network outputs. Note that the input-output sequence of a network must be corrected for the input component with regard for the supplementary input variables introduced in the network component.

2. DERIVATION OF A MONOTONIC DNF SYSTEM FROM THE STG DESCRIPTION OF A SYNCHRONOUS AUTOMATON

Automaton components of a network are subdivided into external components, whose outputs are the network outputs, and internal components, which are other components of the network.

Let us study the STG description of an automaton component (Table 1). Here x_1, x_2, and x_3 are input variables, q are preceding and succeeding internal states, and $y_1, \ldots, y_5$ are output variables.

Instead of the STG description, we study the interval definition of an automaton component describing the same operation as the STG description. For this, let us encode the internal states. For designing a self-testing automaton component, we encode the state by equilibrium codes. Thus, we obtain a list of intervals and their characteristics. The characteristics show the values of transition functions and the outputs at an interval. The interval definition is a description of partial Boolean functions realizing the behavior of the automaton component. Such a description for our example is given Table 2, in which the second row of an interval is of the form

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & z_1 & z_2 & z_3 & z_4 \\ - & 0 & - & 1 & 0 & 0 & 0 \end{array},$$

and its characteristic is expressed by the vector

$$\begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}.$$

Two conjunctions k_1 and k_2 are said to be orthogonal in a variable x_i if x_i is contained without inversion in one conjunction and with inversion in the other. If this condition holds only for one

Table 1

x_1	x_2	x_3	q	q	y_1	y_2	y_3	y_4	y_5
0	–	–	1	1	0	0	0	1	0
–	0	–	1	1	0	0	0	1	0
1	1	–	1	2	1	0	0	1	0
–	–	0	2	2	0	0	1	1	0
–	–	1	2	3	1	0	1	1	0
1	0	–	3	3	0	1	0	0	0
0	–	–	3	4	1	1	0	0	0
–	1	–	3	4	1	1	0	0	0
–	–	0	4	4	0	1	0	0	1
–	–	1	4	1	1	1	0	0	1

Table 2

$x_1\,x_2\,x_3$	$z_1\,z_2\,z_3\,z_4$	$z_1\,z_2\,z_3\,z_4$	$y_1\,y_2\,y_3\,y_4\,y_5$
0 – –	1 0 0 0	1 0 0 0	0 0 0 1 0
– 0 –	1 0 0 0	1 0 0 0	0 0 0 1 0
1 1 –	1 0 0 0	0 1 0 0	1 0 0 1 0
– – 0	0 1 0 0	0 1 0 0	0 0 1 1 0
– – 1	0 1 0 0	0 0 1 0	1 0 1 1 0
1 0 –	0 0 1 0	0 0 1 0	0 1 0 0 0
0 – –	0 0 1 0	0 0 0 1	1 1 0 0 0
– 1 –	0 0 1 0	0 0 0 1	1 1 0 0 0
– – 0	0 0 0 1	0 0 0 1	0 1 0 0 1
– – 1	0 0 0 1	1 0 0 0	1 1 0 0 1

Table 3

$x_1x_2x_3x_4x_5$	$z_1z_2z_3z_4$	$z_1z_2z_3z_4$	$y_1y_2y_3y_4y_5$
0 – – 0 1	1 0 0 0	1 0 0 0	0 0 0 1 0
– 0 – 0 1	1 0 0 0	1 0 0 0	0 0 0 1 0
1 1 – 1 0	1 0 0 0	0 1 0 0	1 0 0 1 0
– – 0 0 1	0 1 0 0	0 1 0 0	0 0 1 1 0
– – 1 1 0	0 1 0 0	0 0 1 0	1 0 1 1 0
1 0 – 0 1	0 0 1 0	0 0 1 0	0 1 0 0 0
0 – – 1 0	0 0 1 0	0 0 0 1	1 1 0 0 0
– 1 – 1 0	0 0 1 0	0 0 0 1	1 1 0 0 0
– – 0 0 1	0 0 0 1	0 0 0 1	0 1 0 0 1
– – 1 1 0	0 0 0 1	1 0 0 0	1 1 0 0 1

variable, then the conjunctions k_1 and k_2 are said to be 1-orthogonal; if the condition holds for two variables, then conjunctions are 2-orthogonal, etc.

If two conjunctions k_1 and k_2 are orthogonal in two variables x_i and x_j such that x_i is contained without inversion in one conjunction and the variables x_j is contained with inversion in the same conjunction, then the conjunctions k_1 and k_2 are said to be inversely 2-orthogonal.

For example, the conjunctions $x_2\overline{x}_3x_4\overline{x}_5$ and $\overline{x}_2\,\overline{x}_3x_5$ are inversely 2-orthogonal in x_2 and x_5. The orthogonality of conjunctions of intervals defined by ternary vectors are defined similarly.

The list of intervals obtained from the STG description through equilibrium coding is partitioned into subsets representing transitions from a given internal state q_i via binary coding q_i. In Table 2, these subsets are separated from one another.

Let U denote the list of all intervals and let U_i be the list of subsets defining a transition from the ith internal state.

Theorem 1. *Intervals of different subsets are inversely 2-orthogonal.*

The proof is implied by the construction of the set U, i.e., coding of states by equilibrium codes.

Intervals with different characteristics inside a set U_i are at least 1-orthogonal, because the automaton is deterministic. To construct a monotonic DNF system, it is necessary that the intervals with different characteristics be inversely 2-orthogonal. Inverse 2-orthogonality is attained by introducing supplementary input variables, i.e., assigning supplementary input components to the ternary vectors that define intervals. Let us describe such an assignment algorithm.

Algorithm 1. (1) Partition the sets U_i into sections $U_{i1}, \ldots, U_{i\tau_1}$ according to their characteristics. Intervals with identical characteristics form a section, $i \in \{1, \ldots, |Q|\}$, where $|Q|$ is the number of different states of the automaton component.

(2) Assign different equilibrium codes (Boolean vectors) to different sections U_{i1}, $1 \in \{1, \ldots, \tau_i\}$.

(3) Assign a suitable code to all intervals of the section U_{i1}.

(4) Repeat steps 1–3 for every subset U_i, minimizing the number of supplementary input variables required for representing by equilibrium codes. Let U^* denote the set of intervals thus obtained.

All intervals with different characteristics in Table 3 constructed for our example are inversely 2-orthogonal. Two supplementary input variables x_4 and x_5 were required for this purpose.

Table 4

$x_1x_2x_3x_4x_5$	$z_1z_2z_3z_4$	$z_1z_2z_3z_4$	$y_1y_2y_3y_4y_5$
$- - - - 1$	$1 - - -$	1 0 0 0	0 0 0 1 0
$1\ 1 - 1 -$	$1 - - -$	0 1 0 0	1 0 0 1 0
$- - - - 1$	$- 1 - -$	0 1 0 0	0 0 1 1 0
$- - 1\ 1 -$	$- 1 - -$	0 0 1 0	1 0 1 1 0
$1 - - - 1$	$- - 1 -$	0 0 1 0	0 1 0 0 0
$- - - 1 -$	$- - 1 -$	0 0 0 1	1 1 0 0 0
$- - - - 1$	$- - - 1$	0 0 0 1	0 1 0 0 1
$- - 1\ 1 -$	$- - - 1$	1 0 0 0	1 1 0 0 1

Theorem 2. *Intervals with different characteristics in the set U^* generated by algorithm 1 are inversely 2-orthogonal.*

The proof of the theorem is implied by the design of the algorithm.

Replacing every 0-component in ternary vectors representing the set U^* by an indeterminate symbol $-$, we obtain a list of monotonic intervals U^{**} and their characteristics. Table 4 is constructed for our example.

Table 4 contains one row less than Table 3, because intervals with identical characteristics are absorbed upon replacement of 0 by the indeterminate symbol $-$.

Theorem 3. *The interval set U^{**} preserves the behavior of an automaton component defined by the STG description.*

The proof of this and succeeding theorems are given in the Appendix.

The set U^{**} of intervals and their characteristics form a monotonic system D—specification for design in a PLM base. Intervals are realized by a matrix of conjunctions of programmable logical matrices and the characteristics of intervals are realized by a matrix of disjunctions of programmable logical matrices.

To use the description of the behavior of component by a monotonic system D instead of the STG description of the component, we must correct input patterns for the interval set U^* (Table 3.)

In the STG description, let us associate inputs with the sequence 000, 001, 111, 111, 101, 000 in the initial state 1. Under the action of an input sequence, the automaton passes through the states 1112334. The corrected sequence (see Table 3) is of the form 00001, 00101, 11110, 11110, 10101, 00010. Precisely this sequence is fed to the component realizing the monotonic system D.

3. PROPERTIES OF A MONOTONIC SYSTEM OF DISJUNCTIVE NORMAL FORMS

Two patterns (vectors) α_i and α_j of values of the variables of the system D are said to congruent if they satisfy the condition $\alpha_i \leq \alpha_j$ or $\alpha_i \geq \alpha_j$ [1].

A pattern α_i is said to precede a pattern α_j and is denoted by $\alpha_i \leq \alpha_j$ if every pair of like components of these patterns satisfy the condition $\alpha_i^t \leq \alpha_j^t$, $t = \{1, \dots, n\}$, where n is the number of variables in the system. For example, the patterns 010010 and 011011 are congruent, and the pattern 010010 precedes the pattern 011011.

Let $D(\alpha)$ be a Boolean vector representing the values of the functions of the system D in the input pattern (vector) α.

(1) If the system D is monotonic and $\alpha_i \leq \alpha_j$, then $D(\alpha_i) \leq D(\alpha_j)$.

Let us illustrate this by an example from Table 4. Let us associate the input variables of the system D (see Table below) with vectors α_i and α_j.

$$\alpha_i = \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}, \quad \alpha_j = \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{array},$$

$$D(\alpha_i) = \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}, \quad D(\alpha_j) = \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array},$$

$$\alpha_i \le \alpha_j, \quad D(\alpha_i) \le D(\alpha_j).$$

Let $D(\alpha_i) = \beta_i$ and $D(\alpha_j) = \beta_j$. Then the vectors β_i and β_j in the example are congruent and $\beta_i \le \beta_j$ (β_i precedes β_j).

The properties of the monotonic system D are the generalization of the properties of a monotonic function.

Let $\widetilde{\alpha}_i$ be a sequence of input patterns. A sequence $\widetilde{\alpha}_i$ is said to precede a sequence $\widetilde{\alpha}_j$ and is denoted by $\widetilde{\alpha}_i \le \widetilde{\alpha}_j$ if the like elements of sequences obey the precedence relation such that every element of the sequence $\widetilde{\alpha}_i$ precedes the corresponding element of the sequence $\widetilde{\alpha}_j$. For example, if $\{\widetilde{\alpha}_i = 010,\ 110,\ 001\}$ and $\{\widetilde{\alpha}_j = 111,\ 110,\ 101\}$, then $\widetilde{\alpha}_i \le \widetilde{\alpha}_j$.

(2) If the system D is monotonic and $\widetilde{\alpha}_i \le \widetilde{\alpha}_j$, then $\widetilde{D}(\widetilde{\alpha}_i) \le \widetilde{D}(\widetilde{\alpha}_j)$ and, consequently, $\widetilde{\beta}_i \le \widetilde{\beta}_j$.

Here $\widetilde{D}(\widetilde{\alpha}_i)$ is the sequence $\widetilde{\beta}_i$ consisting of vectors representing the values of the system D on the vectors of the sequence $\widetilde{\alpha}_i$. For example (Table 4), if

$$\widetilde{\alpha} = \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & z_1 & z_2 & z_3 & z_4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}, \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & z_1 & z_2 & z_3 & z_4 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}, \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array},$$

then

$$\widetilde{D}(\widetilde{\alpha}) = \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}, \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}, \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & y_1 & y_2 & y_3 & y_4 & y_5 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}.$$

The precedence relation is transitive, but not symmetric.

Let us consider two monotonic systems D_r and D_s. Let D_s be derived from D_r via extension of the domain of unit values of the system D_r. The domain of unit values is extended by extending the domain of unit values defining the corresponding system of functions.

In this case, we assume that D_r precedes D_s and is denoted by $D_r \le D_s$.

(3) If $D_r \le D_s$, then $D_r(\alpha) \le D_s(\alpha)$.

(4) If $D_r \le D_s$, then $\widetilde{D}_r(\widetilde{\alpha}) \le \widetilde{D}_s(\widetilde{\alpha})$.

(5) If $\alpha_i \le \alpha_j$, then $D_r(\alpha_i) \le D_s(\alpha_j)$.

(6) If $\widetilde{\alpha}_i \le \widetilde{\alpha}_j$, then $\widetilde{D}_r(\widetilde{\alpha}_i) \le \widetilde{D}_s(\widetilde{\alpha}_j)$.

In the sequel, we use these properties to determine the properties of a monotonic system D in a programmable logical matrix base.

4. PROPERTIES OF A MONOTONIC SYSTEM IN A PLM BASE

A monotonic set of intervals U^{**} (Table 4) along with their characteristics is a monotonic system D of disjunctive normal forms, whose conjunction do not contain any inversions. Every disjunctive normal form of the system contains conjunctions in U^{**}, which are labelled by a unit value of the corresponding component in the characteristics of intervals.

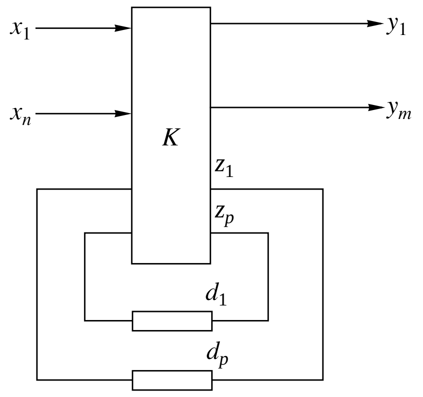

Fig. 1. An internal component of a network.

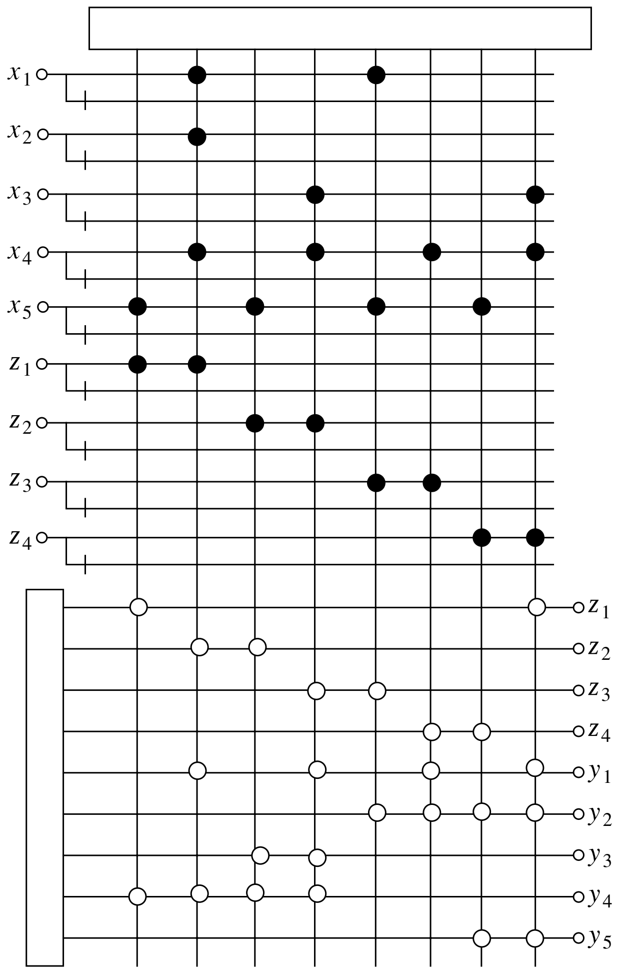

Fig. 2. A realization in PLM base, Table 4.

Figure 1 shows an internal component of a network. Here $x_1, \dots, x_n$, $z_1, \dots, z_p$, and $y_1, \dots, y_m$ are input, internal, and output variables, and $d_1, \dots, d_p$ are feedback delay implemented by D-triggers. The combinational component K is realized by programmable logical matrices. Figure 2 shows a realization in PLM base for our example (Table 4).

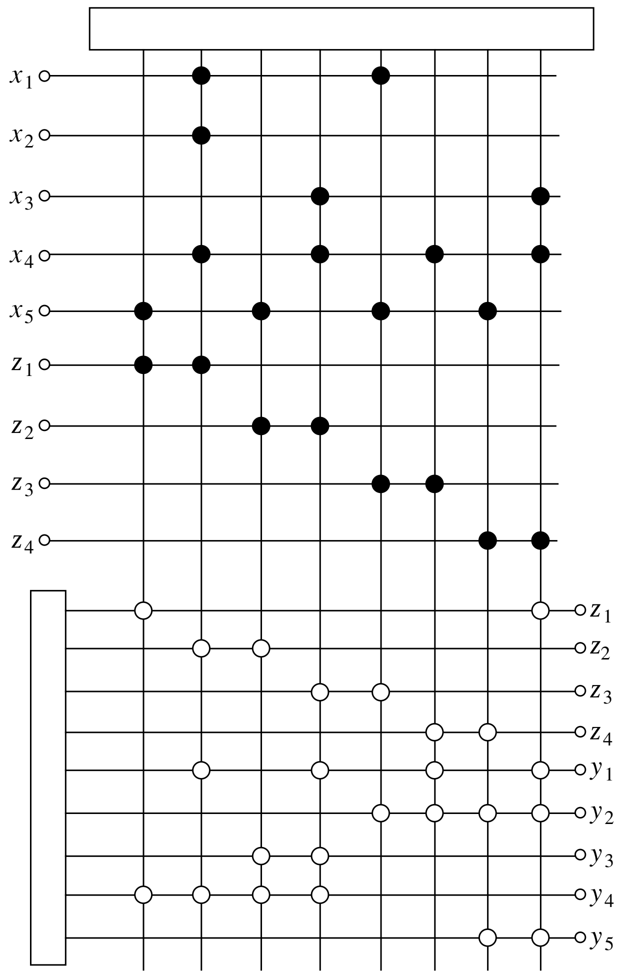

Fig. 3. Realization of Table 4 without decoders.

Since a system D realized by programmable logical matrices is monotonic, there is no need for decoders at conjunction levels [7]. Exclusion of decoders doubles the number of variables in a system D realized by programmable logical matrices. The system D in our example can be realized without any decoder at the conjunction level (Fig. 3).

To realize an STG description (Table 1) without self-testability for the automaton component, we require a pair of buses at conjunction level 2 for state codes and three pairs of buses for the input patterns of the component, i.e., the total number of buses is 10. Nine buses at conjunction levels are required to realize a monotonic system (Table 4).

Let us study solitary faults in programmable logical matrices, i.e., appearance of a connection at a point where it should not occur and absence of a connection at a point where it must exist.

Along with the solitary faults in programmable logical matrices, we shall take account of solitary constant faults at trigger poles and at the input and output poles of the automaton component. Let V denote the set of all faults.

Let us consider the development of faults in the system D realized by programmable logical matrices. Conjunctions of the system D (intervals of U^{**}) are realized at the level of conjunctions of programmable logical matrices without decoders and the characteristics of conjunctions (intervals in the set U^{**}) are realized at the level of disjunctions of programmable logical matrices (Fig. 3).

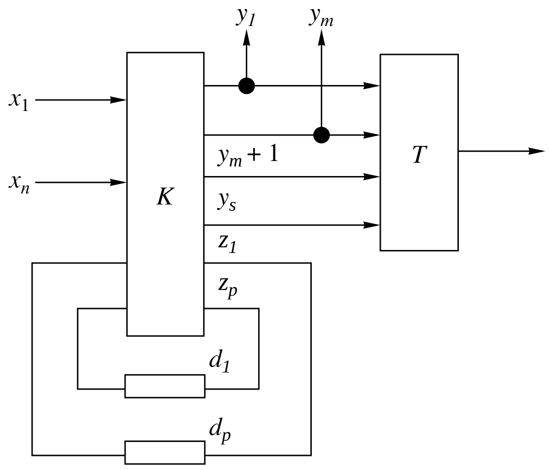

Fig. 4. General view of an external component.

Assertion 1. *As a result of a fault of the set V:*

(a) *a letter in certain conjunctions of D vanishes, but the characteristics of these conjunctions are preserved,*

(b) *a unit component is added to the characteristics of certain conjunctions,*

(c) *certain conjunctions of D vanish,*

(d) *the unit component in the characteristics of certain conjunctions vanishes, and*

(e) *an inversionless variable is added to some conjunction, but the characteristic of this conjunction is preserved.*

Let D^v denote a disjunctive normal form realized by programmable logical matrices in the presence of a fault $v \in V$ and let $D_{\mathrm{a,b}}$ be the disjunctive normal form realized by programmable logical matrices in the presence a fault of the type (a) or (b). The disjunctive normal forms $D_{\mathrm{c,d,e}}$ are defined along similar lines.

Corollary 1.1. $D \leq D_{\mathrm{a,b}}$.

Corollary 1.2. $D_{\mathrm{c,d,e}} \leq D$.

Corollary 1.3. D^v is a monotonic disjunctive normal form.

Corollary 1.4. Every combination of faults of U (multiple faults) generates a monotonic system D^{cri}.

Assertion 2. *Faults generated in the system D by the methods* (a) *and* (b) *are* 1*-monotonic on any test pattern and the faults generated by the methods* (c), (d)*, and* (e) *are* 0*-monotonic on any test pattern of the circuit K.*

Corollary 2.1. Every combination of faults (a multiple fault) generated by the methods (a) and (b) is 1-monotonic on any test pattern of the scheme K.

Corollary 2.2. Every combination of faults (a multiple fault) generated by the methods (c), (d), and (e) is 0-monotonic on any test pattern of the scheme K.

Note that the tester of the external automaton component only "observers" inputs, "not knowing" the changes in the internal states of the component (Fig. 4).

We now examine how a fault of the set V appears when the behavior of the component is under observation.

Note that the circuit realizing the component is provided with outputs such that the output vectors of the automaton component can be encoded by Berger or equilibrium code. Introduction of supplementary outputs "widens" the characteristics of the conjunctions of the system D without changing the conjunctions. The system D remains monotonic.

An input sequence S of the automaton component generates an input sequence $\widetilde{\alpha}$ (for the circuit K) in the space of input and internal variables. A fault $v \in V$ is said to be undetectable on a sequence S if the corresponding output sequence of the component is the same as the output sequence of an operative component. Otherwise, the fault appears on the sequence S.

Theorem 4. *If a fault is* $1(0)$*-monotonically generated in the circuit* K*, it is* $1(0)$*-monotonically generated or undetectable at the outputs of the automaton component.*

Corollary 4.1. Every combination of faults generated in D by the methods (a) and (b) may appear 1-monotonically at the outputs in the operation domain of the automaton component or may not appear in the operation domain.

Corollary 4.2. Every combination of faults generated in D by the methods (c), (d), and (e) may appear 0-monotonically at the outputs in the operation domain of the automaton component or may not appear in the operation domain.

Let us examine the case in which faults of the set V are not generated (not detectable) at the outputs of the automatic component when they are 1(0)-monotonically generated at the outputs of the circuit K. In practice, undetectable faults must be necessarily taken into account, because their presence may change the nature of appearance of the succeeding detectable fault $v \in V$.

Let S be a sequence in the operation domain of the automaton component.

Let $\widetilde{\beta}$ and $\widetilde{\beta}'$ be congruent S output sequences of the circuit K in the absence and presence of faults of the set V. Recall that every subset of the set V transforms the system D into a monotonic system D'. Let $\widetilde{\beta}$ and $\widetilde{\beta}'$ be indistinguishable at the outputs of the automaton component. In this case, no faults appear in the sequence S at the outputs of the automaton component.

If these conditions are satisfied for any sequence S in the operation domain of the component, then a multiple fault of the set V is not detectable.

How does the automaton component respond to a solitary fault $v \in V$?

Theorem 5. *In the presence of undetectable faults of the set* V*, a* $1(0)$*-monotonically generated solitary fault at the outputs of the circuit* K *may appear as a* $1(0)$*-monotonically generated fault at the outputs of the automaton component or remain undetectable.*

Every succeeding solitary fault $v \in V$ appears after the operation domain of the automaton component is full with preceding solitary faults and their multiple fault is undetectable at the outputs of the automaton component.

Let $\widetilde{\alpha}$ be a sequence generated by S and let it be fed to the inputs of the circuit K. A sequence S' is obtained from the sequence S via 1(0)-monotonic transformation of its elements $S \leq S'$ $(S' \leq S)$. Let $\widetilde{\alpha}'$ be the input sequence of the circuit K generated by S'. Since individual elements of the sequence satisfy the condition $D(\alpha) \leq D(\alpha')$ $(D(\alpha') \leq D(\alpha))$, not only the values of input variables in the sequence $\widetilde{\alpha}'$ generated by S', but also the values of internal variables may change 1(0)-monotonically.

Theorem 6. *A* $1(0)$*-monotonic change in the input sequence* S *induces a* $1(0)$*-monotonic change in the output sequence of the automaton component even if the component contains undetectable faults of the set* V*.*

The property of the component described by Theorem 6 is called the "transparency" of the component. It is responsible for the monotonic propagation of the aftereffects of a fault from its point of generation to the circuit outputs.

Using the transparency of the automaton component, including in the presence of undetectable faults, we can design self-testing (synchronous) automaton networks with low cost of optional equipment.

Theorem 7. *The network N^s defines the behavior of the network N.*

5. TRANSFORMATION OF AN AUTOMATON NETWORK INTO A SELF-TESTING NETWORK

Given a network N whose components are synchronous automata defined by STG description, let us construct a self-testing network, connecting testers only to the outputs of external automaton components. For this purpose, let us replace the STG description of every component by a suitable monotonic system D. Let us provide supplementary outputs to external components so that Berger or equilibrium codes can be used at the outputs of these components. External components are also defined by monotonic systems. Every component is realized in a PLM base with D-triggers on feedback lines. Let N^s denote the network thus obtained.

In the previous section, we have shown that a component of an automaton network remains transparent even if its contains undetectable faults. A detectable fault at the outputs of a compo-

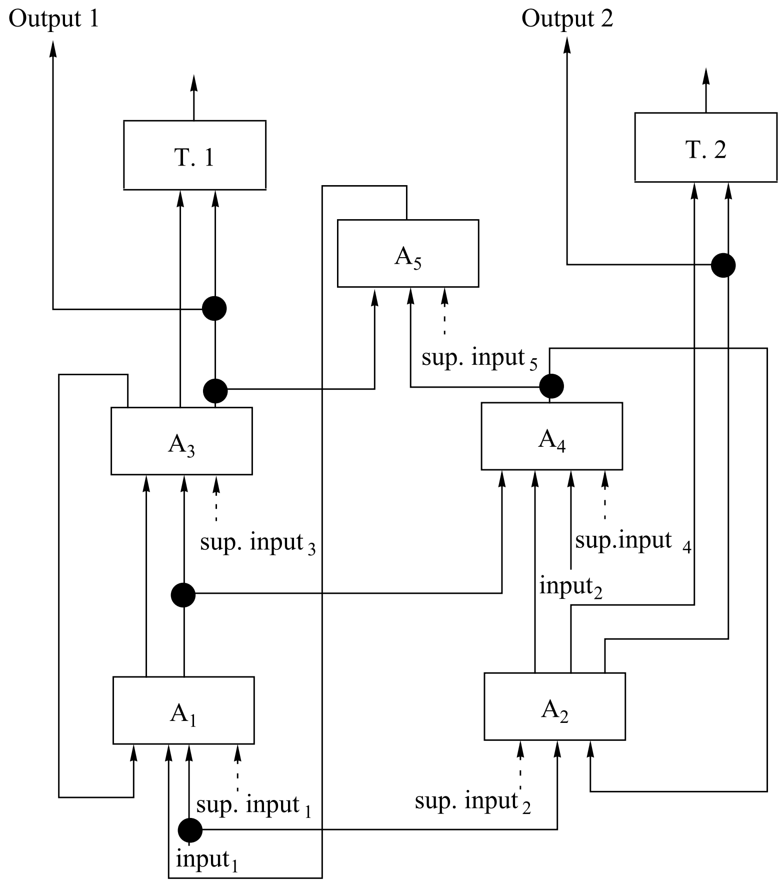

Fig. 5. General view of a self-testing network.

Table 5

	I				II		III		
	n	m	s	ℓ	p	Σ_1	n_{d}	p	Σ_2
BBARA	4	2	10	60	4	8	4	5	13
BBSSE	7	7	16	56	4	11	4	6	17
BBTAS	2	2	6	24	3	5	2	4	8
BEECOUNT	3	4	7	28	3	6	3	5	11
CSE	7	7	16	91	4	11	5	6	18
DK14	3	5	7	56	3	6	3	5	11
DK15	3	5	4	32	2	5	3	4	10
DK16	2	3	27	108	5	7	2	7	11
DK27	1	2	7	14	3	4	1	5	7
DK512	1	3	15	30	4	5	1	6	8
DONFILE	2	1	24	96	5	7	2	7	11
KEYB	7	2	19	170	5	12	7	6	20
LION	2	1	4	11	2	4	2	4	8
LION9	2	1	9	25	4	4	2	5	9
MODULO12	1	1	12	24	4	5	1	6	8
S8	4	1	5	20	3	7	0	4	8
SAND	11	9	32	184	5	16	7	7	25
SHIFTREG	1	1	8	16	3	4	1	5	7
SSE	7	7	16	56	4	11	4	6	17
STYR	9	10	30	166	5	14	6	7	22
TAV.KIS	4	4	4	49	2	6	6	4	14
TBK.KIS	6	3	32	1569	5	11	8	7	21
TRAIN11.KIS	2	1	11	25	4	6	2	6	10
TRAIN4.KIS	2	1	4	14	2	4	2	4	8

nent may not be detectable at the outputs of the network. If a fault is detectable in a network, it is necessarily detectable at the outputs of the component where it is generated.

A solitary fault appears in the network only after the operation domain of the network N^s is full with undetectable faults. A detectable fault appears at the outputs of the network in the operation domain and is detected by testers.

Undetectable faults do not pile up in online self-checking testers [3]. In an online self-checking system consisting of an automaton network and testers, a fault in one of the testers in the presence of undetectable faults in the network, or a fault in the network if the tester is operative is admitted at instant t.

Figure 5 shows a self-testing network. Supplementary input variables of the network components are shown by dotted lines.

Table 5 listing the results of benchmark tests is subdivided into three sections. Section I shows the characteristics of STG descriptions of automaton components. Here n is the number input variables, m is the number of output variables, ℓ is the number of rows in the STG description, and s is the number of (internal) states. Section II shows the characteristics of descriptions of the same automata, whose states are encoded by minimal-length codes. Here p is the number of internal variables required in state coding and Σ_1 is the number of input poles of internal variables of the automaton component that is not a self-testing component. Section III shows the characteristics of descriptions of automaton components by monotonic systems. Here n_{s} is the number of supplementary input variables and Σ_2 is the number of input poles of internal variables of the corresponding monotonic system. As a rule, Σ_2 is not greater than $2\Sigma_1$. Consequently, the same crystal area is required to produce a self-testing automaton component as a component without self-testing facility (exception is the TAV KIS circuit).

6. CONCLUSIONS

An approach is developed to ensuring self-testability of a synchronous automaton network based on the description of a component by a monotonic systems of disjunctive normal forms. Such a description, on the one hand, requires the introduction of supplementary input variables for the component and, on the other hand, aids in using the same crystal area for realizing self-testing internal automaton components as for the conventional realization of these components. Moreover, the use of a monotonic system as a design specification for an automaton component admits the use of testers only at the outputs of external automaton components of the network.

APPENDIX

Proof of Theorem 3. The behavior of a component described by the list of intervals U, along with their characteristics, derived from the state transition graph is the same as that described by the state transition graph. The list U^* defines a behavior that is different from that described by the state transition graph in that the input symbols have elongated codes, i.e., overcoded input symbols. We shall show that U^{**} describes the same behavior as U^*.

Every input pattern that is generated by the state transition graph in the space of input and internal variables belongs to the intervals of U^* with the same characteristic, but not to any interval with other characteristics. Extension of intervals through the replacement of 0-components by the indeterminate symbol "$-$" preserves this property of patterns. Indeed, the pattern under consideration is inversely 2-orthogonal to the intervals in U^* with other characteristics and remains 1-orthogonal to them after extension. This completes the proof of the theorem.

Proof of Assertion 1. Situation (a) arises when a connection in the conjunction matrix vanishes. Situation (e) arises when a connection at the conjunction level appears. Situation (d) arises when a connection at the disjunction level vanishes. Situation (b) arises when a connection at the disjunction level appears. A constant fault 0 at the input or output poles of the D-trigger appears as a "constant 0" fault at the input of the conjunction level and, consequently, the conjunctions containing this variable vanish from the set D, i.e., situation (c) arises.

A constant fault 1 at the input or output poles of the trigger appears as a "constant 1" fault at the input of the conjunction level and, consequently, the character vanishes from all conjunctions of D containing this character, i.e., situation (a) arises.

Similarly, a constant fault at the trigger poles appears as a constant fault at an input pole of the automaton component.

When a constant fault 0 is developed at the output pole of the automaton component, the corresponding unit component vanishes from the characteristics of conjunctions of D, i.e., situation (2) arises. When a constant fault 1 is developed, the corresponding unit component appears in the characteristics of conjunctions of D, i.e., situation (b) arises. This completes the proof.

Proof of Assertion 2. For any input pattern α of the circuit K, we have $D(\alpha) \leq D_{\mathrm{a,b}}(\alpha)$. If α is a test pattern, then the fault is developed 1-monotonically. This completes the proof.

Proof of Theorem 4. Consider the faults generated by the methods (a) and (b). In this case, $D \leq D_{\mathrm{a,b}}$. Let $\alpha(t)$ be an element of the sequence $\widetilde{\alpha}$ fed to the circuit K at instant t. Then $D(\alpha(t)) \leq D_{\mathrm{a,b}}(\alpha(t))$. Therefore, additional units may appear at the outputs of the circuit K, including feedback lines, in the presence of a fault, i.e., $\alpha(t+1) \leq \alpha^*(t+1)$ at the succeeding instant. Here $\alpha^*(t+1)$ is obtained from $\alpha(t)$ in the presence of a fault of the type (a) or (b). Consequently, the input sequence S of the automaton component in the presence of a fault generates a sequence $\widetilde{\alpha}^*$ such that $\widetilde{\alpha} \leq \widetilde{\alpha}^*$. Therefore, $\widetilde{D}(\widetilde{\alpha}) \leq \widetilde{D}_{\mathrm{a,b}}(\widetilde{\alpha}^*)$ (property 6 of monotonic systems).

If a fault is developed in the sequence S in the operation domain of the component, then it is 1-monotonically generated. It may not appear in the operation domain of the component.

Let us consider the faults generated by the methods (c), (d), and (e). In this case, $D_{\mathrm{c,d,e}} \le D$, $D_{\mathrm{c,d,e}}(\alpha(t)) \le D(\alpha(t))$, $\alpha^*(t+1) \le \alpha(t+1)$, $\widetilde{\alpha}^* \le \widetilde{\alpha}$. Therefore, $\widetilde{D}_{\mathrm{c,d,e}}(\widetilde{\alpha}^*) \le \widetilde{D}(\widetilde{\alpha})$.

If a fault is generated in the sequence S in the operation domain of the component, then it is generated 0-monotonically. It may not be detectable at the operation domain. This completes the proof of the theorem.

Proof of Theorem 5. Let us consider the input pattern $\alpha(t)'$ of the circuit K that is generated by the sequence S and a multiple undetectable fault. This sequence either differs from $\alpha(t)$ in the absence of a fault, or it coincides with it. But in either case, its corresponding output patterns $\beta(t)'$ and $\beta(t)$ do not differ in output variables of the automaton component.

If a new solitary fault $v \in V$ of the type (a) and (b) is generated at instant t, then $D'(\alpha'(t)) \le D''(\alpha'(t))$, where D'' is a monotonic system generated by D' and an additional solitary fault and D' is a monotonic system generated by the undetectable fault. Consequently, $\alpha'(t+1) \le \alpha''(t+1)$, where $\alpha''(t+1)$ is the input pattern of the circuit K at instant $t+1$ in the presence of an additional fault: $D'(\alpha'(t+1)) \le D''(\alpha''(t+1))$ (property 5 of monotonic systems).

An additional fault $v \in V$ may be generated 1-monotonically at the outputs of the automaton components or may not appear at all.

The fault $v \in V$ of the type (c), (d), and (e) generated at instant t induces the situation $D''(\alpha'(t)) \le D'(\alpha'(t))$. Consequently, $\alpha''(t+1) \le \alpha'(t+1)$ and $D''(\alpha''(t+1)) \le D'(\alpha'(t+1))$. The additional fault $v \in V$ may be generated 0-monotonically at the outputs of the automaton component or may not appear at all. This completes the proof of the theorem.

Proof of Theorem 6. Since $\widetilde{\alpha} \le \widetilde{\alpha}'$ ($\widetilde{\alpha}' \le \widetilde{\alpha}$), we have $\widetilde{D}(\widetilde{\alpha}) \le \widetilde{D}(\widetilde{\alpha}')$ ($\widetilde{D}(\widetilde{\alpha}') \le \widetilde{D}(\widetilde{\alpha})$). If there are undetectable faults, then the circuit K realizes the monotonic system D^{cri}. Consequently, $\widetilde{D}^{\mathrm{cri}}(\widetilde{\alpha}) \le \widetilde{D}^{\mathrm{cri}}(\widetilde{\alpha}')$ ($\widetilde{D}^{\mathrm{cri}}(\widetilde{\alpha}') \le \widetilde{D}^{\mathrm{cri}}(\widetilde{\alpha})$). This completes the proof of the theorem.

Proof of Theorem 7. Let us consider the complete state α of the network N at instant t. It is the union of complete input states (states of inputs and feedback line) of the component. By a complete state of a network, we mean the internal states that are encoded by equilibrium codes (Table 2). Upon coding the states, we obtain the same description of the behavior of the component as the description defined by the state transition graph.

Let us associate a state α with a state γ formed by the union of characteristics of interval sets U generated by complete input states in α. The state γ defines the outputs of the automaton network and network state at instant $t+1$. In describing the automaton component by interval sets U^* and then by U^{**} (monotonic systems), we indeed elongate the input components of automaton components in the vector α (by generating the vector α'), leaving the vector γ unchanged. In other words, the same vector γ corresponds to the vectors α and α'.

Consequently, the behavior of the automaton network does not change upon replacement of the STG descriptions of components by the sets U^{**} (monotonic systems). This completes the proof.

REFERENCES

1. Baranov, S., *Logic Synthesis for Control Automata*, Dordrecht: Kluwer, 1994.

2. Yablonskii, S.V., *Vvedenie v diskretnuyu matematiku* (Introduction to Discrete Mathematics), Moscow: Nauka, 1979.

3. Levin, I. and Karpovsky, M., Online Self-Checking of Microprogram Control Units, *4th IEEE Int. On-Line Testing Workshop, Compendium of Papers*, Capri, 1998, pp. 152–156.

4. Busaba, F. and Lala, P., Combinational Circuit Design for Single and Unidirectional Multibit Error, *JETTA*, 1994, no. 5, pp. 19–28.

5. Goessel, M. and Sogomonyan, E.S., Code Disjoint Self-Parity Combinational Circuits for Self-Testing, Concurrent Fault Detection and Parity Scan Design, *Proc. 12th IEEE VLSI Test Symp.*, 1994, pp. 151–157.

6. Matrosova, A. and Ostanin, S., Self-Checking Synchronous FSM Network Design, *4th IEEE Int. On-Line Testing Workshop, Compendium of Papers*, Capri, 1998, pp. 162–166.

7. Moto-oka, T., Takaka, H., *et al.*, *Computers on VLSI*, Tokyo: Iwanami Shoteu, 1984. Translated under the title *Komp'yutery na SBIS*, Moscow: Mir, 1988.

This paper was recommended for publication by P.P. Parkhomenko, a member of the Editorial Board