DESIGN OF SELF-CHECKING DISCRETE DEVICES BASED ON BOOLEAN SIGNALS CORRECTION AND COMPOSITION OF CONSTANT-WEIGHT CODES OF THE "1-OUT-OF-4" AND "3-OUT-OF-4" TYPES. PART I: The Design Method with Conversion of All Signals from the Object under Diagnosis

D. V. Efanov* and Y. I. Yelina**

*,**Peter the Great Saint Petersburg Polytechnic University, St. Petersburg, Russia
*Solomenko Institute of Transport Problems, Russian Academy of Sciences, St. Petersburg, Russia

*✉ TrES-4b@yandex.ru, **✉ eseniya-elina@mail.ru

Abstract. It is proposed to use the composition of constant-weight codes of the "1-out-of-4" and "3-out-of-4" types in the design of self-checking discrete devices based on Boolean signals correction. For this composition of constant-weight codes, a simple and compact self-checking checker can be implemented, requiring only four test combinations for a complete check. The structure of a self-checking discrete device is described. The conversion of all four signals from an object under diagnosis is considered when designing a concurrent error-detection circuit for the object. The simplest algorithm for building a self-checking concurrent error-detection circuit based on the Boolean signals correction and the composition of 1-out-of-4 and 3-out-of-4 codes is developed. The results of experiments with combinational benchmarks are provided. They demonstrate the advantage of the proposed approach over the well-known duplication method and the one involving the 1-out-of-4 code together with Boolean signals correction, in terms of structural redundancy of the final self-checking discrete device. The method developed in this paper is promising for self-checking discrete device design with different element bases in various fields of application.

Keywords: self-checking discrete device, concurrent error-detection circuit, Boolean signals correction, composition of constant-weight codes, 1-out-of-4 and 3-out-of-4 codes, structural redundancy, controllable structure.

INTRODUCTION

When designing self-checking discrete devices, common methods are based on introducing redundancy not into the original object itself (the object under diagnosis) but into a special concurrent error-detection (CED) circuit [1–3]. Such a circuit is built taking into account the characteristics of the object under diagnosis and the selected diagnostic attribute for detecting faults and errors in computations. In essence, a working diagnosis system is implemented for a given object: based on the results of computing its functions, this system indirectly identifies the presence/absence of faults [4]. CED circuits are implemented as self-checking, which eliminates the problem of "watch dog" when the developer intends to ensure operability control for the CED circuit itself. Totally self-checking CED circuits are self-testing and protected from faults [5]. According to the first property, for all faults of a CED circuit from a given class, there exists at least one set of argument values applied to its inputs to identify a control error signal. The second property states that when any fault occurs in a CED circuit from a given class, either correct values are computed at its outputs, or an error signal is identified.

There are two main approaches to building CED circuits, each generating a variety of methods. The first (classical) approach has been known since the

middle of the last century [6]. Within this approach, data signals are formed at the parallel outputs of an object under diagnosis; when applying each set of argument values to the inputs, these signals are supplemented by check signals in the CED circuit to establish a certain correspondence between the data and check signals if the object is operable. This is often done using the diagnostic attribute of the belonging of codewords, formed by data and check signals, to a given uniform block code [7–9]. Object's faults during its operation cause errors in computations, which violate the correspondence between the values of the data and check signals and, in turn, are identified by the CED circuit. The second approach has been known since the 1990s–2000s [10–13]. In contrast to supplementing data signals with check signals, this approach converts all or part of the signals so that the required codeword with specified diagnostic properties is identified for each set of argument values.

Both approaches have advantages and drawbacks. For example, the approach based on signal supplementation directly considers the error detection properties of uniform block codes and makes it possible to use special circuitry methods surely detect certain types of errors in CED circuits [14, 15]. However, for a given element basis and method for building CED circuit components, its structure is unique, and the developer cannot regulate the performance characteristics of the final self-checking discrete device (e.g., structural redundancy, controllability, power consumption, etc.). For each set of argument values, the second approach (based on Boolean signals correction, BSC) allows selecting a codeword from a certain set to convert the binary vector from the outputs of an object under diagnosis; this feature provides wide possibilities for building various CED circuits even for a given element base and implementation methods [16]. In addition, optimization problems can be solved for a particular performance indicator of CED circuits. However, due to BSC for the vector at the object's outputs, well-known circuit engineering techniques cannot be applied to cover certain types of errors in CED circuits. Alternative approaches are required, such as the selection of subsets of outputs with special control properties (e.g., those on which errors with a given multiplicity d (single, double, triple, etc.) cannot occur) [17]. In all these cases, there are two options for building CED circuits: the first is the complete coverage of errors at the object's outputs, caused by its internal structure faults; and the second is the manifestation of faults at least on one set of argument values [18, 19].

The simplest way to implement CED circuits using the second approach is based on the use of inseparable block codes. They include constant-weight codes [20], Plotkin (Hadamard) codes [21], Borden codes [22], and various other compositions of constant-weight codes [23]. A distinctive feature of methods with such codes is that when organizing CED circuits, it is easy to use circuit engineering techniques to eliminate certain types of errors in codewords, whereas for separable codes, it is also necessary to consider potential simultaneous distortions of both data and check symbols [24].

Research by the authors has shown that, in addition to traditional constant-weight codes, there is a special composition of these codes formed by combining codewords belonging to the "1-out-of-4" and "3-out-of-4" types [23] (further referred to as 1-out-of-4 and 3-out-of-4 codes, respectively). A checker with a simple and compact structure can be built for it, requiring only four test combinations for a complete check. This is the minimum cardinality of the set of codewords forming a test for self-checking checkers [5].

This study is devoted to describing CED circuit design methods based on BSC with compositions of 1-out-of-4 and 3-out-of-4 codes. It consists of two parts: in the first, we present the CED circuit structure based on BSC with conversion of signals from all outputs of an object under diagnosis and the simplest algorithm for building a self-checking CED circuit; the second part will provide methods for reducing structural redundancy by utilizing the properties of the composition of 1-out-of-4 and 3-out-of-4 codes and decreasing the number of conversion elements in the CED circuit.

1. CED CIRCUIT BASED ON THE COMPOSITION OF 1-OUT-OF-4 AND 3-OUT-OF-4 CODES

The structure of a CED circuit based on the composition of 1-out-of-4 and 3-out-of-4 codes is shown in Fig. 1. It is built for a set of four outputs of an object under diagnosis. The output set of cardinality 4 is chosen due to the length of the code vector of the composition of 1-out-of-4 and 3-out-of-4 codes.

An object under diagnosis is the combinational part of a discrete device, or simply a combinational discrete device $F(X)$ that computes the values of Boolean functions $f_1(X)$, $f_2(X)$, $f_3(X)$, and $f_4(X)$ under a given set $\langle X\rangle = \langle x_t\ x_{t-1} \dots x_2\ x_1\rangle$ of argument values supplied to the inputs. (Hereinafter, this set is supposed to be complete.) Thus, a Boolean vector $\langle f_4(X)\ f_3(X)\ f_2(X)\ f_1(X)\rangle$ is formed for each set of argument values. A special CED circuit is organized to check the correctness of computations. It consists of three functional blocks: the block $G(X)$ for computing the values of correction functions, the signal correction block (SCB) itself, and a *totally self-checking checker* (TSC) for the composition of 1-out-of-4 and 3-out-of-4 codes.

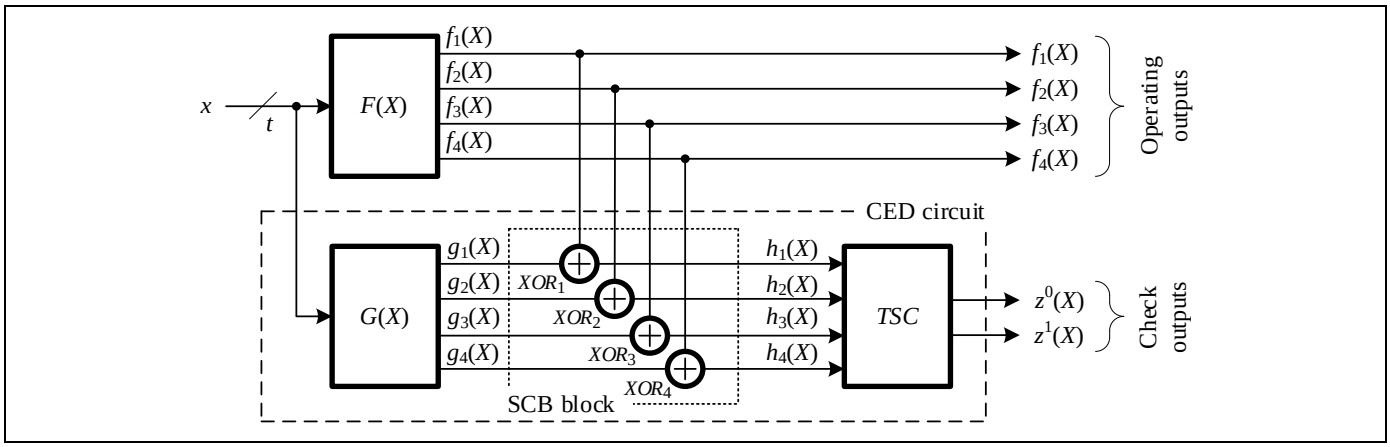

Fig. 1. The structural diagram of a CED circuit.

The block $G(X)$ is intended to compute the values of the functions $g_1(X)$, $g_2(X)$, $g_3(X)$, and $g_4(X)$ for correcting signals from the object $F(X)$ when identical sets of argument values are supplied to the inputs of both blocks. Each signal correction function implements the following conversion in the CED circuit:

$$h_i(X) = f_i(X) \oplus g_i(X),\ i = \overline{1,4}, \qquad (1)$$

where $h_i(X)$ is the corrected value at the SCB output obtained under a particular set of argument values supplied to the inputs.

Two-input XORs are used to correct signals. They allow any value 0 (1) to be converted to any of the values 0 (1).[1] Thus, for each set of argument values, the code vector $<f_4(X)\ f_3(X)\ f_2(X)\ f_1(X)>$ can be converted into the code vector $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ with specified diagnostic properties. The structure shown in Fig. 1 performs conversion to a codeword belonging to the composition of 1-out-of-4 and 3-out-of-4 codes. The TSC block is installed to verify the belonging of the codeword $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ to the given composition. It is equipped with two outputs, $z^0(X)$ and $z^1(X)$, which operate in two-rail logic: the combinations <01> and <10> indicate the absence of errors in computations whereas the combinations <00> and <11> the presence of an error (and, indirectly, the occurrence of a fault in the object under diagnosis).

Figure 2 demonstrates the structure of the simplest checker for the composition of 1-out-of-4 and 3-out-of-4 codes.

[1] The equivalence function implemented by an XNOR gate has a similar property. It could also be used in the CED circuit design based on BSC. Other elementary Boolean functions can be applied to BSC only with several restrictions.

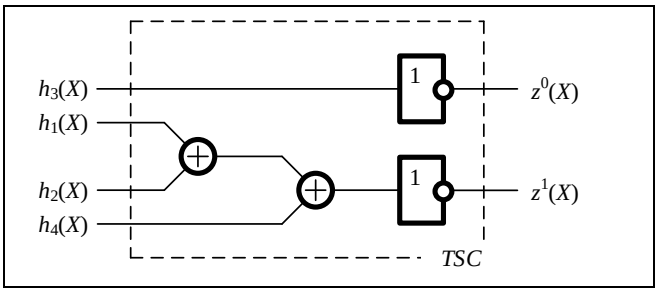

Fig. 2. The structural diagram of a checker for the composition of 1-out-of-4 and 3-out-of-4 codes.

The checking test T_{che} for the checker in Fig. 2 contains combinations from the following set:

$$T_{\text{che}} = \{1000, 0010, 1101, 0111\} \cup \{0001, 1011, 0100, 1110\}. \qquad (2)$$

When designing CED circuits, it suffices to form at least one subset of the set (2). In this case, a complete check of TSC can be performed during the operation of the self-checking device.

Thus, the structure in Fig. 1 is self-checking.

2. THE SIMPLEST ALGORITHM FOR BUILDING A CED CIRCUIT BASED ON THE COMPOSITION OF 1-OUT-OF-4 AND 3-OUT-OF-4 CODES

The conversion (1) is performed on each set of argument values. The vector $<f_4(X)\ f_3(X)\ f_2(X)\ f_1(X)>$ can be converted into the code vector $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ in eight different variants since the set of codewords for the composition of the 1-out-of-4 and 3-out-of-4 codes has cardinality $C_4^1 + C_4^3 = 8$. Hence, for t arguments, the conversion (1) can be performed in 8^{2^t} variants. Due to the need

to form the set T_{che} for the checker, we have $8^{2^t} - 3$ variants. Even in the case $t = 3$, this gives 16 777 213 variants to design self-checking CED circuits. Among them, the best variant can be selected, e.g., by the minimum structural redundancy criterion of a self-checking discrete device.

Meanwhile, let us propose a simple algorithm for building a CED circuit based on BSC and the composition of 1-out-of-4 and 3-out-of-4 codes, which forms the set T_{che} for the checker, but requires post-verification of the test combinations formed for XOR gates. Assume that all functions of an object under diagnosis are completely defined, and all 2^t sets of argument values are supplied to the inputs. Note also that at the initial stage, the structures of the BSC and TSC blocks are known, and the logic of $F(X)$ is specified; however, the values of the functions implemented by the block $G(X)$ are not set. The main goal of design is to obtain the values of these functions for each set of argument values and to form checking tests for the BSC and TSC blocks.

Algorithm 1. *CED circuit design for four outputs of the object under diagnosis:*

1. Determine $\delta = \dfrac{2^t}{8} = 2^{t-3}$, which is the number of each of the words used in the composition of constant-weight codes $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$. As a result, one builds a CED circuit in which the codewords of the composition of 1-out-of-4 and 3-out-of-4 codes will be formed uniformly at the TSC inputs.

2. Consider the sets of argument values in lexicographic order, from the set with decimal equivalent 0 to the set with decimal equivalent $2^t - 1$. On the sets of argument values with decimal equivalents $0\ldots(2^{t-3} - 1)$, redefine the bits of the vectors $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ to the codeword <0001>; on the sets with numbers $2^{t-3}\ldots(2\cdot 2^{t-3} - 1)$, to the codeword <0010>; on the sets with numbers $2\cdot 2^{t-3}\ldots(3\cdot 2^{t-3} - 1)$, to the codeword <0100>; on the sets with numbers $3\cdot 2^{t-3}\ldots(4\cdot 2^{t-3} - 1)$, to the codeword <1000>; on the sets with numbers $4\cdot 2^{t-3}\ldots(5\cdot 2^{t-3} - 1)$, to the codeword <1110>; on the sets with numbers $5\cdot 2^{t-3}\ldots(6\cdot 2^{t-3} - 1)$, to the codeword <1101>; on the sets with numbers $6\cdot 2^{t-3}\ldots(7\cdot 2^{t-3} - 1)$, to the codeword <1011>; finally, on the sets with numbers $7\cdot 2^{t-3}\ldots(8\cdot 2^{t-3} - 1)$, the codeword <0111>. This step of the algorithm forms each of the codewords for the composition of 1-out-of-4 and 3-out-of-4 codes at the TSC inputs exactly δ times.

3. Determine the values of the functions $g_i(X)$ based on the known values of the functions $h_i(X)$ for each set of argument values:

$$\begin{gathered} g_i(X) = f_i(X) \oplus h_i(X) \\ \Leftrightarrow h_i(X) = f_i(X) \oplus g_i(X),\ i = \overline{1,\ 4}. \end{gathered} \qquad (3)$$

4. Verify the formation of the checking test for each XOR gate in the CED circuit: under the canonical realization of XOR, the checking test includes all four working combinations {00, 01, 10, 11} [25], and each of them shall be formed on at least one set of argument values. If tests are formed for all SCB gates, proceed to Step 5 of the algorithm; otherwise, redefine the bit values in the vector $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ on a selected number of the sets of argument values in accordance with the method described in [26].

5. Optimize the functions $g_4(X)$, $g_3(X)$, $g_2(X)$, and $g_1(X)$ [27].

6. Design a self-checking discrete device in the selected element basis.

We pay the reader's attention to Step 2 of the algorithm, which involves all eight codewords belonging to the given composition. However, in view of the expression (2), this step can be changed: the vectors $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ can be redefined to only four codewords of the given composition, which will ensure the complete check of the checker.

Example 1. Let us build a CED circuit for a device whose operating logic is described by the first nine columns of Table 1.

Following Step 1 of Algorithm 1, we determine the number $\delta = 2^{4-3} = 2$, characterizing the number of each of the words used in the composition of 1-out-of-4 and 3-out-of-4 codes. Then, according to Step 2 of Algorithm 1, we sequentially consider the sets of argument values and fix the following codewords of the composition of 1-out-of-4 and 3-out-of-4 codes in the vector $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$: on sets 0 and 1, the word <0001>; on sets 2 and 3, the word <0010>; on sets 4 and 5, the word <0100>; on sets 6 and 7, the word <1000>; on sets 8 and 9, the word <1110>; on sets 10 and 11, the word <1101>; on sets 12 and 13, the word <1011>; finally, on sets 14 and 15, the word <0111> (see the columns corresponding to the bits of the vector $<h_4(X)\ h_3(X)\ h_2(X)\ h_1(X)>$ in Table 1). Using Step 3 of Algorithm 1 and formula (3), we obtain the values of the functions implemented at the outputs of the block $G(X)$ for each set of argument values (see the columns corresponding to the bits of the vector $<g_4(X)\ g_3(X)\ g_2(X)\ g_1(X)>$ in Table 1). Following Step 4 of Algorithm 1, we determine whether checking tests are formed for each of the gates XOR_4, XOR_3, XOR_2, and XOR_1. All combinations formed at their inputs are given in the corresponding columns of Table 1. Analysis of the combinations arriving at the inputs of each correction element shows that a checking test is formed for each of them.

Table 1

The operation of a device with four outputs and the CED circuit for it

No.	The sets of argument values				The values at the outputs of the device $F(X)$				The values at the outputs of the SCB block				The values at the outputs of the block $G(X)$				Test combinations of the SCB block			
	x_4	x_3	x_2	x_1	$f_4(X)$	$f_3(X)$	$f_2(X)$	$f_1(X)$	$h_4(X)$	$h_3(X)$	$h_2(X)$	$h_1(X)$	$g_4(X)$	$g_3(X)$	$g_2(X)$	$g_1(X)$	XOR_4	XOR_3	XOR_2	XOR_1
0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	1	11	11	00	01
1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	1	00	00	11	01
2	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	10	00	11	00
3	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	01	00	00	00
4	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	00	10	00	0
5	0	1	0	1	0	0	1	0	0	1	0	0	0	1	1	0	00	01	11	00
6	0	1	1	0	0	1	0	0	1	0	0	0	0	1	1	0	00	11	01	00
7	0	1	1	1	0	0	0	1	1	0	0	0	0	0	1	1	00	00	01	11
8	1	0	0	0	1	0	1	0	1	1	1	0	1	1	1	0	11	01	11	00
9	1	0	0	1	1	0	0	0	1	1	1	0	1	1	0	0	11	01	00	00
10	1	0	1	0	0	1	1	0	1	1	0	1	1	1	1	0	01	11	11	00
11	1	0	1	1	0	0	0	1	1	1	0	1	1	1	0	0	01	01	00	10
12	1	1	0	0	1	0	1	1	1	0	1	1	0	0	0	0	10	00	10	10
13	1	1	0	1	1	0	1	0	1	0	1	1	0	0	0	1	10	00	10	01
14	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	11	10	10	10
15	1	1	1	1	0	1	0	1	0	1	1	1	0	0	1	0	00	10	01	10

Next, Steps 5 and 6 of Algorithm 1 should be carried out, but they appear to be trivial: the functions $g_4(X)$, $g_3(X)$, $g_2(X)$, and $g_1(X)$ are optimized, and a self-checking discrete device is implemented in the selected element basis. ♦

A CED circuit for a device with $n > 4$ outputs is built as follows.

Algorithm 2. *CED circuit design for an object under diagnosis with n > 4 outputs:*

1. Order the device outputs and divide them into $q = \left\lceil \frac{n}{4} \right\rceil$ subsets. (The symbol $\lceil ... \rceil$ denotes the ceiling of a corresponding value.) If n(mod4) = 0, each of the q subsets will contain unique outputs; otherwise, the last subset will contain n(mod4) unique outputs and 4 – n(mod4) ones already used in q – 1 subsets.

2. Design the CED circuit according to Algorithm 1 for each of the q subsets of outputs. To reduce redundancy, the individual blocks $G(X)$ of each CED circuit shall be built together.

3. Connect the outputs of q CED circuits to the inputs of the self-checking comparator qTRC1, which compresses q two-rail signals into one and is based on q – 1 elementary *two-rail checkers* (TRC) [28–30].

Example 2. Provide the structure of a self-checking discrete device for organizing a CED circuit for the initial device with $n = 10$ outputs.

According to Step 1 of Algorithm 2, we order the outputs and divide them into the following number of subsets: $q = \left\lceil \frac{10}{4} \right\rceil = 3$. The first two subsets are disjoint: I – $\{f_1(X), f_2(X), f_3(X), f_4(X)\}$, and II – $\{f_5(X), f_6(X), f_7(X), f_8(X)\}$. Since 10(mod4) = 2, the last subset includes two outputs from the second subset and two previously unused outputs: III – $\{f_7(X), f_8(X), f_9(X), f_{10}(X)\}$.

Figure 3 shows the structure of a self-checking discrete device in Example 2. To implement it, three CED circuits are allocated with separate SCB blocks and checkers. The check outputs of individual CED circuits are connected to the inputs of a self-checking comparator implemented on two TRC blocks. Since the outputs $f_7(X)$ and $f_8(X)$ are used twice, the corresponding functions in the CED circuit are marked by "*" and "**," respectively. ♦

The main advantage is that Algorithms 1 and 2 provide the simplest possible design of CED circuits for initial objects under diagnosis, and the procedures implemented in these algorithms can be easily automated for further use in computer-aided design systems for self-checking discrete devices.

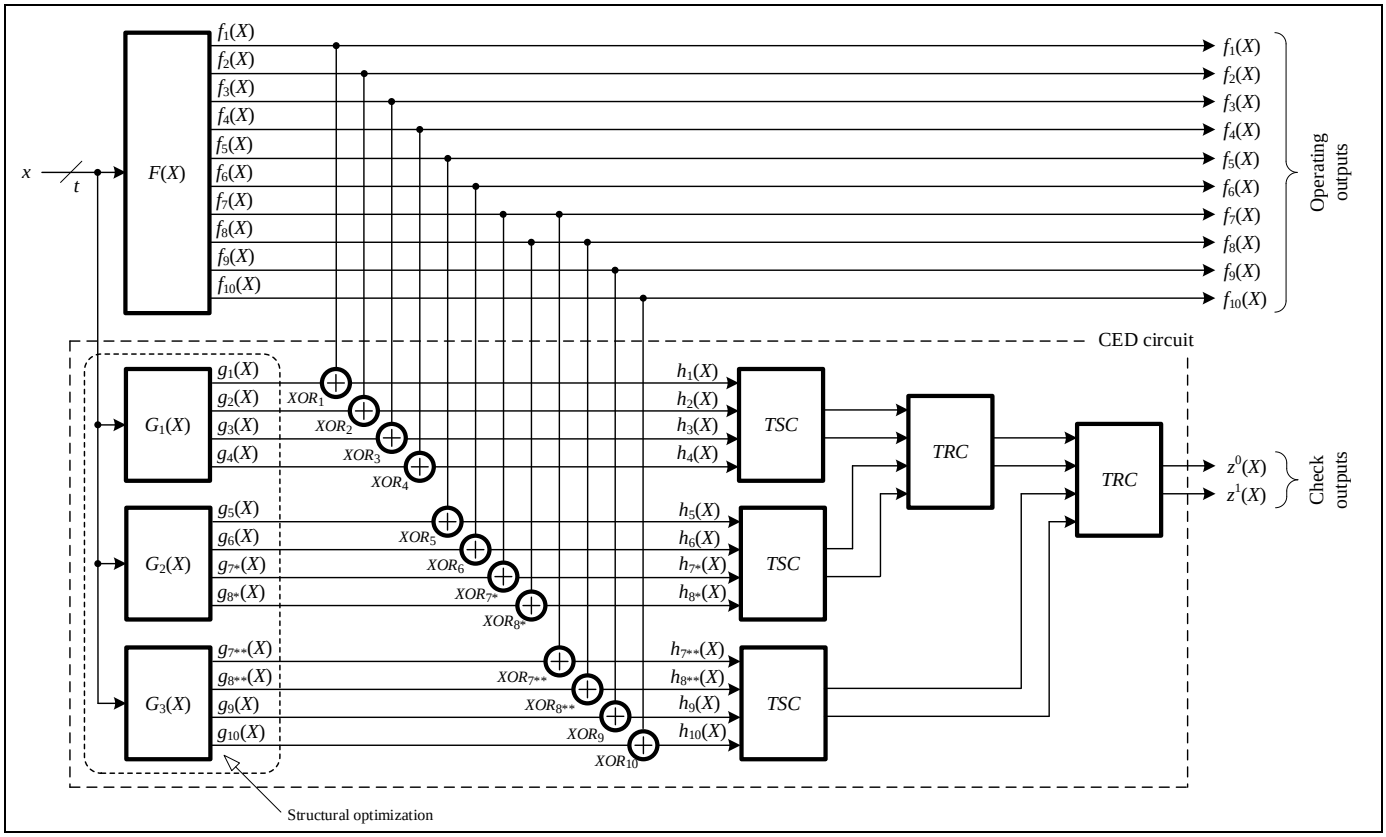

Fig. 3. The structural diagram of the self-checking discrete device in Example 2.

3. EXPERIMENTAL RESULTS

An important task of this study is to confirm experimentally the efficiency of the structure proposed (Fig. 1), as well as Algorithms 1 and 2, in the design of self-checking discrete devices for test combinational circuits (benchmarks) from MCNC Benchmarks [31, 32]. During the experiments, structural redundancy indicators were estimated for self-checking discrete devices designed for each benchmark. For comparison, two methods were also used with the same total number of outputs for the blocks $G(X)$ as for the object under diagnosis. The first method was classical duplication: all blocks $G(X)$ were actually replaced by a single copy of the object or by a device computing equivalent Boolean functions [6]. (Nowadays, this method is widely used to build self-checking implementations of discrete devices [33].) The second method was the one based on BSC and the use of the 1-out-of-4 code with correction of all signals from the object [34].

CED circuits were designed for the selected benchmarks by three methods. Combinational benchmarks in the *.pla format were taken. In this format, a circuit is specified by an interval description of Boolean functions (as a compressed truth table representing a ternary matrix with elementary conjunctions) [35]. The processing techniques for benchmarks, as well as the algorithms developed, were automated and realized in *DM Coding*[2] to obtain descriptions in the *.pla format for the functional blocks of CED circuits. Next, the well-known SIS interpreter [36, 37] and the *stdcell2_2.genlib* library of functional elements were used to design self-checking devices and determine some of their parameters. During the design, the *full_simplify* optimization procedure was applied for the blocks $G(X)$. (The other blocks of CED circuits were typical.) This procedure serves to optimize a system of Boolean functions using binary decision diagrams (BDDs) [38]. Then, the *map -s* procedure was carried out to perform the structural synthesis of devices in the given library of functional elements and obtain their main parameters. For comparison, an additional parameter—the area occupied by the device on a chip—was considered to characterize the implemen-

[2] DM Coding is a software module developed by the authors of this paper jointly with Cand Sci. (Eng.) V.V. Dmitriev. This module is expandable and intended for setting up experiments with novel fault diagnosis methods. Written in C#, it implements certain functions for analyzing benchmarks with the algorithms being developed, including the functional description of CED circuit blocks built by various methods.

tation complexity of a device (in conditional units of the element library). For each CED circuit component, the implementation complexity indicators were determined, including the final values of the implementation complexity indicator of a self-checking device (by analogy with the method described in [39]). For duplication, this indicator is denoted by L_D; for the method based on BSC and the 1-out-of-4 code, by $L_{1/4}$; for the novel method (proposed above), by $L_{1/4+3/4}$. The following relative indicators were calculated for comparing the three methods with each other:

$$\mu = \frac{L_{1/4+3/4}}{L_D} \cdot 100\%, \quad (4)$$

$$\eta = \frac{L_{1/4+3/4}}{L_{1/4}} \cdot 100\%. \quad (5)$$

The indicators (4) and (5) characterize the self-checking device designed by the novel method in comparison with those designed by duplication and BSC with the 1-out-of-4 code. If $\mu > 100\%$ ($\eta > 100\%$), then the novel method is more efficient than duplication (BSC with the 1-out-of-4 code, respectively). Also, for a convenient efficiency analysis of the novel method, the following indicators (the relative gain in its complexity) were calculated:

$$\Delta\mu = \left(1 - \frac{L_{1/4+3/4}}{L_D}\right) \cdot 100\,\%,$$

$$\Delta\eta = \left(1 - \frac{L_{1/4+3/4}}{L_{1/4}}\right) \cdot 100\,\%.$$

A value $\Delta\mu > 0\%$ ($\Delta\eta > 0\%$) indicates a gain in the complexity of the novel method compared to duplication (BSC with the 1-out-of-4 code, respectively).

Table 2 summarizes the results of experiments for several benchmarks. For each benchmark, the values of the implementation complexity indicators $L_{F(X)}$, L_D, $L_{1/4}$, and $L_{1/4+3/4}$ are provided. For the methods based on BSC, the values of the implementation complexity indicator $L_{G(X)}$ of the block $G(X)$ are also given. For circuits with $n \geq 5$ outputs, the block $G(X)$ was obtained by combining separate blocks for computing the values of signal correction functions for each of the check and design subcircuits into a single circuit. These indicators can be compared for different methods based on BSC in order to assess the impact of the complexity indicator of the block for computing correction functions on the final value of the complexity indicator of the self-checking discrete device. Separately, the bar charts in Fig. 4 show the values of $\Delta\mu$ and $\Delta\eta$.

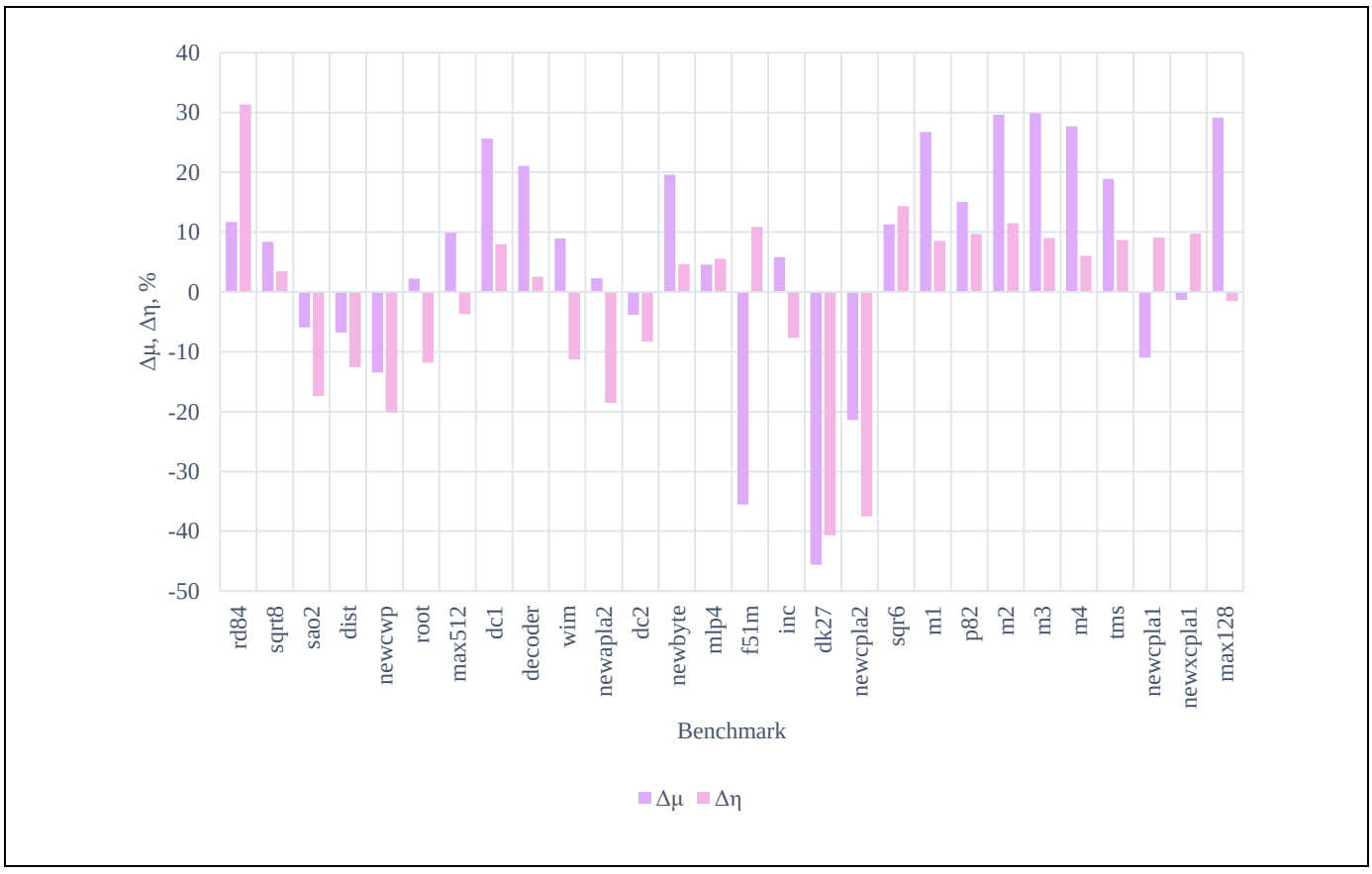

Fig. 4. The values of Δμ and Δη for different benchmarks.

Table 2

Experimental results

Benchmark	n	q	$L_{F(X)}$	L_D	The novel method		The method from [34]		μ	η	Δμ	Δη
					$L_{G(X)}$	$L_{1/4+3/4}$	$L_{G(X)}$	$L_{1/4}$				
rd84	4	1	4912	10464	4056	9240	8128	13456	88.303	68.668	11.697	31.332
sqrt8	4	1	1160	2960	1280	2712	1232	2808	91.622	96.581	8.378	3.419
sao2	4	1	2992	6624	3752	7016	2568	5976	105.918	117.403	–5.918	–17.403
dist	5	2	6968	14784	8080	15784	6448	14024	106.764	112.55	–6.764	–12.55
newcwp	5	2	440	1728	784	1960	584	1632	113.426	120.098	–13.426	–20.098
root	5	2	3496	7840	3432	7664	2752	6856	97.755	111.785	2.245	–11.785
max512	6	2	9632	20320	7944	18312	7224	17656	90.118	103.715	9.882	–3.715
dc1	7	2	976	3216	680	2392	632	2600	74.378	92	25.622	8
dekoder	7	2	736	2736	688	2160	488	2216	78.947	97.473	21.053	2.527
wim	7	2	712	2688	1000	2448	496	2200	91.071	111.273	8.929	–11.273
newapla2	7	2	600	2464	1072	2408	480	2032	97.727	118.504	2.273	–18.504
dc2	7	2	2424	6112	3184	6344	2440	5856	103.796	108.333	–3.796	–8.333
newbyte	8	2	592	2656	808	2136	384	2240	80.422	95.357	19.578	4.643
mlp4	8	2	7224	15920	7232	15192	7520	16088	95.427	94.431	4.573	5.569
f51m	8	2	2272	6016	5144	8152	5528	9144	135.505	89.151	–35.505	10.849
Inc	9	3	2376	6432	2480	6056	1712	5624	94.154	107.681	5.846	–7.681
dk27	9	3	528	2736	2256	3984	768	2832	145.614	140.678	–45.614	–40.678
newcpla2	10	3	1896	5680	3800	6896	1392	5016	121.408	137.48	–21.408	–37.48
sqr6	12	3	2648	7600	2896	6744	2672	7872	88.737	85.671	11.263	14.329
m1	12	3	3064	8432	1912	6176	1096	6752	73.245	91.469	26.755	8.531
p82	14	4	2368	7456	2304	6336	1712	7016	84.979	90.308	15.021	9.692
m2	16	4	10096	23328	4656	16416	4288	18544	70.37	88.525	29.63	11.475
m3	16	4	13464	30064	5952	21080	5536	23160	70.117	91.019	29.883	8.981
m4	16	4	18704	40544	8952	29320	8344	31208	72.316	93.95	27.684	6.05
tms	16	4	6784	16704	5096	13544	3928	14832	81.082	91.316	18.918	8.684
newcpla1	16	4	2520	8176	4888	9072	3304	9984	110.959	90.865	–10.959	9.135
newxcpla1	23	6	3760	12112	5920	12272	3216	13600	101.321	90.235	–1.321	9.765
max128	24	6	20192	45184	9232	32016	4528	31536	70.857	101.522	29.143	–1.522

According to Table 2 and Fig. 4, we draw the following conclusions. For 9 of the 28 benchmarks, duplication is more efficient. However, for most benchmarks (19, approximately 68%), the novel method yields self-checking discrete devices with smaller implementation complexity. Moreover, the gain for some benchmarks is $\Delta\mu > 20\%$. Compared to the method described in [34], the novel one is more effective for 16 benchmarks (approximately 57%). The improvement in the indicator $\Delta\eta$ does not exceed 10% for most benchmarks. This is explained by the similarity between the novel method and the one from [34] in terms of the number of signals corrected. The effect is achieved mainly due to the gain in the indicator $L_{G(X)}$ when applying one or another method. For three benchmarks (see Fig. 4), a significant loss is obtained in the implementation complexity indicator: over 20% for both $\Delta\mu$ and $\Delta\eta$. This result is due to the complexity of the Boolean functions implemented by the blocks $F(X)$ and $G(X)$ [40]. In some cases, the complexity of Boolean functions implemented by the block $G(X)$ is significantly higher than that of the ones implemented by the block $F(X)$. (The reader can compare the data in the columns $L_{F(X)}$ and $L_{G(X)}$ row by row.) Nevertheless, for most benchmarks, the novel method demonstrates a positive effect, which suggests its practical applicability.

CONCLUSIONS

When designing self-checking discrete devices based on BSC, the composition of 1-out-of-4 and 3-out-of-4 codes can be efficiently used. The checker for this code has a simple and compact structure, and four combinations are sufficient for a complete check. As a result, it is rather easy to build self-checking CED circuits for discrete devices, e.g., using the algorithms presented above.

According to the structural redundancy analysis of the experimental results obtained using the software modules and products, the method proposed in this paper has advantages over some known design methods of self-checking discrete devices. Moreover, as will be demonstrated in part II of the study, the properties of the composition of 1-out-of-4 and 3-out-of-4 codes can be utilized to reduce the number of signals converted from the object under diagnosis and, thereby, decrease the number of outputs of the blocks $G(X)$. Such a structural modification procedure for CED circuits shown in Fig. 1 will provide an even greater gain in terms of the implementation complexity of self-checking discrete devices compared to well-known methods. Note that other metrics, synthesizers, and sets of functional elements can also be used to determine the features and characteristics of the novel method.

The advantage of using BSC and the composition of 1-out-of-4 and 3-out-of-4 codes is a large number of variants to design CED circuits: one can solve related optimization problems and build self-checking discrete devices, e.g., with the lowest hardware cost. A drawback of the novel method is the need to consider the number of combinations of distortions at the outputs of an object under diagnosis when selecting quadruples among them. There are two approaches for building a CED circuit: the classical one with full coverage of any errors at the object's outputs, and the one with coverage of faults occurring at least on one set of argument values. Also, it is possible to use the first approach for implementing a CED circuit by covering any errors at the object's outputs with an increased number of the checked subsets of outputs (as compared to the minimum required for the novel method), but with an advantage over other methods in terms of implementation complexity. The possibility of such an implementation has been demonstrated by some experimental results, particularly by the above analysis of the gains in complexity indicators.

REFERENCES

1. Sogomonyan, E.S. and Slabakov, E.V., *Samoproveryaemye ustroistva i otkazoustoichivye sistemy* (Self-Checking Devices and Fault-Tolerant Systems), Moscow: Radio i Svyaz', 1989. (In Russian.)
2. Mitra, S. and McCluskey, E.J., Which Concurrent Error Detection Scheme to Choose?, *Proceedings of International Test Conference*, Atlantic City, 2000, pp. 985–994. DOI: 10.1109/TEST.2000.894311
3. Göessel, M., Ocheretny, V., Sogomonyan, E., and Marienfeld, D., *New Methods of Concurrent Checking*, 1st ed., Dordrecht: Springer Science+Business Media, 2008.
4. Drozd, A.V., An Untraditional View on Operational Diagnostics of Computing Devices, *Control Sciences*, 2008, no. 2, pp. 48–56. (In Russian.)
5. Sapozhnikov, V.V. and Sapozhnikov, Vl.V., *Samoproveryaemye diskretnye ustroistva* (Self-Checking Discrete Devices), St. Petersburg: Energoatomizdat, 1992. (In Russian.)
6. Goessel, M. and Graf, S., *Error Detection Circuits*, London: McGraw-Hill, 1994.
7. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., *Kody Khemminga v sistemakh funktsional'nogo kontrolya logicheskikh ustroistv* (Hamming Codes in Functional Control Systems of Logical Devices), St. Petersburg: Nauka, 2018. (In Russian.)
8. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., *Kody s summirovaniem dlya sistem tekhnicheskogo diagnostirovaniya. Tom 1: Klassicheskie kody Bergera i ikh modifikatsii* (Sum Codes for Technical Diagnosis Systems. Vol. 1: Classical Berger Codes and Their Modifications), Moscow: Nauka, 2020. (In Russian.)
9. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., *Kody s summirovaniem dlya sistem tekhnicheskogo diag-*

nostirovaniya. Tom 2: Vzveshennye kody s summirovaniem, (Sum Codes for Technical Diagnosis Systems. Vol. 2: Weight-Based Codes), Moscow: Nauka, 2021. (In Russian.)
10. Goessel, M., Saposhnikov, V., Saposhnikov, Vl., and Dmitriev, A., A New Method for Concurrent Checking by Use of a 1-Out-Of-4 Code, *Proceedings of 6th IEEE International On-Line Testing Workshop*, Palma de Mallorca, Spain, 2000, pp. 147–152. DOI: 10.1109/OLT.2000.856627
11. Dmitriev, A., Saposhnikov, V., Saposhnikov, V., and Goessel, M., New Self-Dual Circuits for Error Detection and Testing, *VLSI Design*, 2000, vol. 11, no. 1, pp. 1–21. DOI: 10.1155/2000/84720
12. Sapozhnikov, V.V., Sapozhnikov, Vl.V., Dmitriev, A.V., et al., Organization of Functional Control of Combinational Circuits Using the Boolean Complement Method, *Electronic Modeling*, 2002, vol. 24, no. 6, pp. 52–66. (In Russian.)
13. Gessel, M., Morozov, A.V., Sapozhnikov, V.V., and Sapozhnikov, Vl.V., Logic Complement, a New Method of Checking the Combinational Circuits, *Automation and Remote Control*, 2003, vol. 64, no. 1, pp. 153–161.
14. Saposhnikov, V.V., Morosov, A., Saposhnikov, Vl.V., and Göessel, M., A New Design Method for Self-Checking Unidirectional Combinational Circuits, *Journal of Electronic Testing: Theory and Applications*, 1998, vol. 12, no. 1-2, pp. 41–53. DOI: 10.1023/A:1008257118423
15. Matrosova, A.Yu. and Mitrofanov, E.V., Delay Testable Sequential Circuit Design, *Tomsk State University Journal of Control and Computer Science*, 2013, no. 2 (23), pp. 140–147. (In Russian.)
16. Efanov, D.V. and Yelina, Y.I., Investigation of Ways of Synthesizing Concurrent Error-Detection Circuits Based on Boolean Signal Correction Using Uniform Separable Codes, *Russian Microelectronics*, 2024, vol. 53, no. 5, pp. 471–482. DOI: 10.1134/S1063739724600456
17. Efanov, D.V., Synthesis of Self-Checking Computing Devices Based on a Complete System of Special Groups of the Diagnostic Object Outputs, *Journal of Instrument Engineering*, 2023, vol. 66, no. 5, pp. 355–372. DOI: 10.17586/0021-3454-2023-66-5-355-372 (In Russian.)
18. Goessel, M. and Sogomonyan, E.S., Design of Self-Testing and Self-Checking Combinational Circuits with Weakly Independent Outputs, *Automation and Remote Control*, 1992, vol. 53, no. 8, pt. 2, pp. 1264–1272.
19. Sogomonyan, E.S. and Gössel, M., Design of Self-Testing and On-Line Fault Detection Combinational Circuits with Weakly Independent Outputs, *Journal of Electronic Testing: Theory and Applications*, 1993, vol. 4, no. 4, pp. 267–281. DOI: 10.1007/BF00971975
20. Freiman, C.V., Optimal Error Detection Codes for Completely Asymmetric Binary Channels, *Information and Control*, 1962, vol. 5, no. 1, pp. 64–71. DOI: 10.1016/S0019-9958(62)90223-1
21. MacWilliams, F.J. and Sloane, N.J.A., *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.
22. Borden, J.M., Optimal Asymmetric Error Detecting Codes, *Information and Control*, 1982, vol. 53, no. 1-2, pp. 66–73. DOI: 10.1016/S0019-9958(82)91125-1
23. Efanov, D.V., Compositions of Two Constant-Weight Codes with Orthogonal Combinations over All Bits for Self-Checking Discrete Device Design, *Control Sciences*, 2025, no. 3, pp. 41–61.
24. Efanov, D., Osadchy, G., and Zueva, M., Special Aspects of Errors Definition via Sum Codes within Embedded Control Schemas Being Realized by Means of Boolean Complement Method, *Proceedings of 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2021)*, Cracow, Poland, 2021, vol. 1, pp. 424–431. DOI: 10.1109/IDAACS53288.2021.9660837
25. Aksenova, G.P., Necessary and Sufficient Conditions for the Synthesis of Completely Testable Modulo 2 Convolution Circuits, *Automation and Remote Control*, 1979, vol. 40, no. 9, pp. 1362–1369.
26. Efanov, D.V., Self-Checking Combinational Devices Synthesis Based on the Boolean Signal Correction Method Using Bose—Lin Codes, *Information Technologies*, 2023, vol. 29, no. 10, pp. 503–511. DOI: 10.17587/it.29.503-511 (In Russian.)
27. Zakrevskii, A.D., Pottosin, Yu.V., and Cheremisinova, L.D., *Logicheskie osnovy proektirovaniya diskretnykh ustroistv* (Logical Foundations of Discrete Device Design), Moscow: Fizmatlit, 2007. (In Russian.)
28. Parkhomenko, P.P. and Sogomonyan, E.S., *Osnovy tekhnicheskoi diagnostiki. Tom 2: Optimizatsiya algoritmov diagnostirovaniya, apparaturnye sredstva* (Foundations of Technical Diagnosis. Vol. 2: Optimization of Diagnostic Algorithms, Hardware Means), Moscow: Energoatomizdat, 1981. (In Russian.)
29. Lala, P.K., *Self-Checking and Fault-Tolerant Digital Design*, San Francisco: Morgan Kaufmann Publishers, 2001.
30. Chioktour, V. and Kakarountas, A., Adaptive BIST for Concurrent On-Line Testing on Combinational Circuits, *Electronics*, 2022, vol. 19, no. 11, pp. 1–20. DOI: 10.3390/electronics11193193
31. McElvain, K., *IWLS'93 Benchmark Set. Version 4.0*. Distributed as a Part of IWLS'93 Benchmark Set, 1993.
32. *Collection of Digital Design Benchmarks*. URL: https://ddd.fit.cvut.cz/www/prj/Benchmarks/ (Accessed August 26, 2025.)
33. Hahanov, V., Litvinova, E., Chumachenko, S., et al., Vector Logic Modeling Self-Tested Circuits, *Proceedings of the 21st IEEE East-West Design & Test Symposium (EWDTS'2025)*, Tbilisi, Georgia, 2025. DOI: 10.1109/EWDTS67441.2025.11303706
34. Efanov, D.V., Using the "1-Out-Of-4" Constant-Weight Code in the Synthesis of Self-Checking Concurrent Error-Detection Circuit Based on Boolean Signal Correction, *Tomsk State University Journal of Control and Computer Science*, 2025, no. 72, pp. 114–133. DOI: 10.17223/19988605/72/12 (In Russian.)
35. Zakrevskij, A.D. and Toropov, N.R., Minimization of Boolean Functions of Many Variables – Iterative Method and Program Realization, *Applied Discrete Mathematics*, 2009, no. 1, pp. 5–14. (In Russian.)
36. Sentovich, E.M., Singh, K.J., and Moon, C., Sequential Circuit Design Using Synthesis and Optimization, *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers & Processors*, Cambridge, 1992, pp. 328–333. DOI: 10.1109/ICCD.1992.276282
37. *SIS: A System for Sequential Circuit Synthesis*, Sentovich, E.M., Singh, K.J., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P.R., Brayton, R.K., and Sangiovanni-Vincentelli, A., Berkeley: Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, 1992.
38. Bibilo, P.N., *Binarnye diagrammy reshenii v logicheskom proektirovanii* (Binary Decision Diagrams in Logical Design), Moscow: LENAND, 2024. (In Russian.)
39. Efanov, D.V. and Yelina, Y.I., Design of Self-Checking Digital Devices with Boolean Signals Correction Using Weight-Based Bose–Lin Codes, *Control Sciences*, 2024, no. 4, pp. 22–36. DOI: http://doi.org/10.25728/pu.2024.4.3

40.Korshunov, A.D., Computational Complexity of Boolean Functions, *Russian Math. Surveys*, 2012, vol. 67, no. 1, pp. 93–165. https://doi.org/10.1070/RM2012v067n01ABEH004777

This paper was recommended for publication by L.Yu. Filimonyuk, a member of the Editorial Board.

Received September 14, 2025, and revised November 27, 2025. Accepted November 27, 2025.

Author information

Efanov, Dmitry Viktorovich. Dr. Sci. (Eng.), Peter the Great Saint Petersburg Polytechnic University, St. Petersburg, Russia; Solomenko Institute of Transport Problems, Russian Academy of Sciences, St. Petersburg, Russia
✉ TrES-4b@yandex.ru
ORCID iD: https://orcid.org/0000-0002-4563-6411

Yelina, Yeseniya Igorevna. Postgraduate, Peter the Great Saint Petersburg Polytechnic University, St. Petersburg, Russia
✉ eseniya-elina@mail.ru
ORCID iD: https://orcid.org/0009-0004-4167-3591

Cite this paper
Efanov, D.V. and Yelina, Y.I., Design of Self-Checking Discrete Devices Based on Boolean Signals Correction and Composition of Constant-Weight Codes of the “1-Out-Of-4” and “3-Out-Of-4” Types. Part I: The Design Method with Conversion of All Signals from the Object under Diagnosis. *Control Sciences* 1, 30–40 (2026).

Original Russian Text © Efanov, D.V., Yelina, Y.I., 2026, published in *Problemy Upravleniya*, 2026, no. 1, pp. 34–46.

Translated into English by *Alexander Yu. Mazurov*,
Cand. Sci. (Phys.–Math.),
Trapeznikov Institute of Control Sciences,
Russian Academy of Sciences, Moscow, Russia
✉ alexander.mazurov08@gmail.com