ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2026, Vol. 87, No. 5, pp. 465–481.
© The Author(s), 2026 published by Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, 2026.
Russian Text © The Author(s), 2026, published in Avtomatika i Telemekhanika, 2026, No. 5, pp. 108–130.

ROBUST, ADAPTIVE, AND NETWORK CONTROL

Design of Self-Checking Discrete Devices Based on Boolean Signal Correction with Weighted Sum Codes in the Residue Ring of a Given Modulus

D. V. Efanov[*,a,**] **and Y. I. Yelina**[*,b]

[*] *Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia*
[**] *Solomenko Institute of Transport Problems, Russian Academy of Sciences, St. Petersburg, Russia*
e-mail: [a] *TrES-4b@yandex.ru,* [b] *eseniya-elina@mail.ru*

Received June 16, 2025
Revised November 17, 2025
Accepted December 10, 2025

Abstract—It is proposed to design concurrent error-detection (CED) circuits for discrete automation and computing devices using Boolean signal correction (BSC) with weighted sum codes. Within this approach, a CED circuit corrects signals from all outputs of an object under diagnosis and involves a particular subset of codewords of a preselected weighted sum code. An algorithm is developed for selecting codewords used to design a CED circuit based on BSC; this algorithm allows choosing the best variant to cover faults at the object's outputs and ensures the self-checking property of the circuit. The features of organizing CED circuits based on the above method are shown.

Keywords: concurrent error-detection (CED) circuit, Boolean signals correction, weighted sum code, self-checking device, computation control at device outputs

DOI: 10.7868/S1608303226050065

1. INTRODUCTION

Many units and nodes of automation and computing systems must be self-checking for the timely detection of faults in their structures and mitigation of their manifestations [1]. This is especially important in critical systems, where a fault (error) may cause an accident or even a catastrophe.

In the design of self-checking discrete devices, methods of information theory, coding, and Boolean function theory are widely used [2–6]. Considering the properties of error-detecting and error-correcting codes makes it possible to design devices with given reliability characteristics and fault detection capabilities by manifestations in the form of signal distortions at specially selected checkpoints or operating outputs [7, 8].

This paper is devoted to further developing the theory of self-checking discrete device design through implementing external fault diagnosis means in the form of concurrent error-detection (CED) circuits using the properties of binary redundant codes for error detection only (without error correction). As such codes, we consider a wide class of weighted sum codes in the residue ring of a given modulus [9]. According to the authors' research, these codes can be effectively applied in the design of CED circuits based on Boolean signal correction (BSC); the idea to use such signal correction for building self-checking discrete devices was proposed in [10, 11]. With BSC, when each set of argument values is supplied to the inputs of a CED circuit, the signals from an object under diagnosis are converted into signals treated by the circuit as codewords of a preselected code. (This approach is in contrast to the conventional one, where the object's signals are supplemented with check signals without any conversion [12, 13].) Due to the properties of weighted sum codes, it is possible to design self-checking devices based on BSC and choose implementation options with the best values of the structural redundancy and testability indicators of CED circuit gates.

2. PROBLEM STATEMENT

Existing CED circuit design methods involve the error-detection properties of uniform block codes and the diagnostic properties of Boolean functions. In CED circuit design, the general error-detection properties of uniform block codes and Boolean functions are used. In fact, a CED circuit "inherits" the detecting properties of the redundant code (or a particular class of Boolean functions) selected at the design stage. It becomes impossible to consider in a CED circuit the structural features of objects under diagnosis (the configurations of gates and their connections with each other, as well as with inputs and outputs of the device, which affects the multiplicity and types of allowed errors at their outputs caused by internal structure faults). Therefore, two approaches emerge for further development of CED circuit design methods. The first is to construct a new redundant code by taking the object's structure and a given fault model into account. Such a problem was solved in [14]. The second approach is to use a particular subset of the set of codewords of some uniform block code in order to orient the selected codewords to cover faults at the object's outputs with certain properties. Research shows that this is possible using BSC. Let us formulate the following problem.

Consider a given combinational object under diagnosis $F(X)$, which computes the values of Boolean functions $f_1(X), f_2(X), \ldots, f_n(X)$ when supplying the sets of argument values $< x_t x_{t-1} \ldots x_2 x_1 > = < X >$ at its inputs. It is required to build a self-checking discrete device based on BSC and a wide class of weighted sum codes with a particular subset of the set of its codewords. (This class covers codes well known in the theory of self-checking discrete device design.) In addition, it is necessary to develop a method for extracting a particular subset of codewords of a weighted sum code whose codewords have desired diagnostic properties (e.g., capabilities to detect faults with given multiplicities or to implement Boolean functions from special classes in a CED circuit), a method for modifying the encoder of this code to work with a particular subset of its codewords, and a CED circuit design algorithm based on BSC and a particular subset of codewords of this code.

Note that in a special case, the object under diagnosis can be a subcircuit of the device $F(X)$ whose outputs have certain properties of error propagation under internal structure faults, being controllable for the chosen CED circuit organization using a particular subset of codewords of a weighted sum code.

3. THE ORGANIZATION STRUCTURE OF A CED CIRCUIT

In a CED circuit based on BSC, the signals coming from an object under diagnosis $F(X)$ are converted by a signal correction block (SCB) so as to obtain a codeword of a given code (Fig. 1). For correction, two-input mod 2 adders (XORs) are used; they uniformly generate 0 and 1 when all sets of argument values are supplied to the inputs and can correct any signal to 0 and 1. The first inputs of the gates receive signals $f_1(X), \ldots, f_n(X)$ from the object $F(X)$, and the second inputs receive signals from the block $G(X)$, which computes the values of correction functions $g_1(X), \ldots, g_n(X)$, predetermined for each set $< X >$. The conversion is given by

$$h_i(X) = f_i(X) \oplus g_i(X), \quad i = \overline{1,n}. \tag{1}$$

Initially, at the CED circuit design stage, the values of the functions $g_1(X), \ldots, g_n(X)$ are not specified. The main task is to obtain their values on each set of argument values. For organizing computation control, the type of functions computed at SCB outputs (the functions $h_1(X), \ldots, h_n(X)$) is important. According to (1), for known values of the functions $f_1(X), \ldots, f_n(X)$, the values of the functions $g_1(X), \ldots, g_n(X)$ are uniquely determined by the values of the functions $h_1(X), \ldots, h_n(X)$:

$$g_i(X) = f_i(X) \oplus h_i(X), \quad i = \overline{1,n}. \tag{2}$$

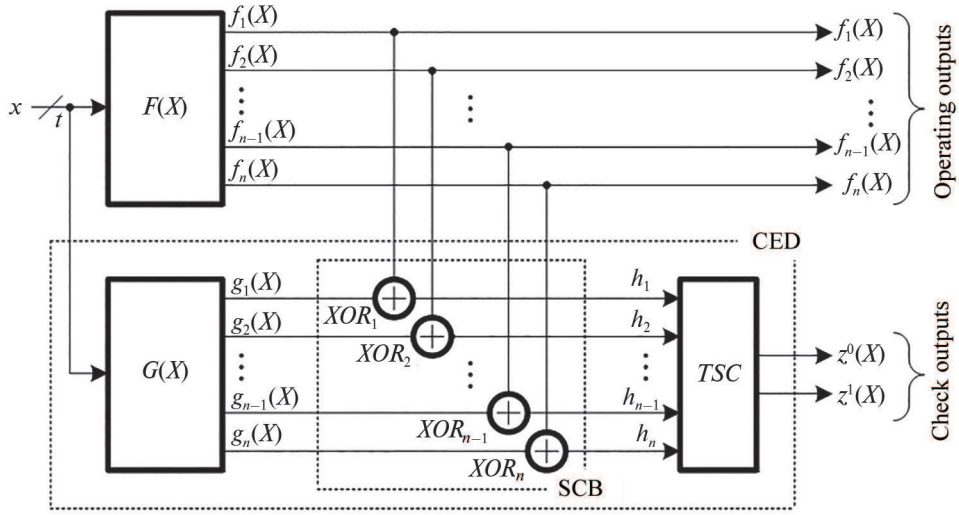

Fig. 1. The block diagram of a CED circuit based on BSC.

The methods for determining the values of the functions $h_1(X), \ldots, h_n(X)$ depend on the binary redundant code chosen for computation control. These methods are categorized as heuristic (a complement on each set of argument values considered sequentially, under the existing constraints on this procedure due to the requirements for generating test combinations for SCB gates and the checker [15]) and functional (an initially established relationship between the values of the functions $g_1(X), \ldots, g_n(X)$ and $f_1(X), \ldots, f_n(X)$ before the CED circuit design stage [16]). Thus, when supplying each set of argument values $< x_t x_{t-1} \ldots x_2 x_1 >$ to the device inputs, a codeword belonging to the chosen code is formed at SCB outputs; this is checked in the CED circuit by the totally self-checking checker (TSC) [17]. As the "watch dog," this checker has two outputs $z^0(X)$ and $z^1(X)$. In the absence of errors at its inputs and internal faults, the checker generates the two-rail signal $< 01 >$ or $< 10 >$. Violation of the two-rail nature indicates the presence of errors in computations and, indirectly, the presence of faults in the object under diagnosis or the CED circuit itself.

The difference between the above approaches to CED circuit design lies precisely in the fact that the conventional approach implies no correction but complements the signals generated at the outputs of the block $F(X)$ with check signals computed by an additional block in the CED circuit. Therefore, the methods widely used in the conventional approach take into account special properties of signal propagation to the device outputs under faults, e.g., the unidirectionality of occurring errors [18–20] or unidirectionality and their asymmetry (inequality in the number of corrupted 0s and 1s) [21, 22]. For the block diagram presented in Fig. 1, it is impossible to consider the type of error occurring at the object's output directly; therefore, computation control methods based on isolating subsets of independent [23] and r-independent outputs (outputs at which errors with multiplicities $d < r$ are allowed [24]) can be effective here.

Within the conventional approach to CED circuit design for a selected error-detecting code, there exists only one option to form the functions computed by control devices. (In other words, it is impossible to build various CED circuits, except for different implementations of the additional block for computing check functions and the checker.) In the case of BSC for a selected error-detecting code, there are numerous variants to build a CED circuit, determined only by the resulting codeword, i.e., the conversion of a particular codeword $< f_n(X) f_{n-1}(X) \ldots f_2(X) f_1(X) > = < F >$ on each set of argument values $< x_t x_{t-1} \ldots x_2 x_1 >$. Owing to the large number of variants to build a CED circuit, the self-checking property can be ensured in practice by choosing an appropriate

implementation. For example, BSC allows building self-checking devices in several cases where this property cannot be achieved by conventional approaches with duplication and parity checking [11].

The following peculiar constraints are imposed on the design procedure with BSC:

(1) For a complete check of *TSC*, when supplying sets of argument values to the inputs, it is necessary to form all codewords of a given code that belong to the set of combinations making up the check test. (This is not always the complete set of codewords of the code under consideration, which is determined by the method for building *TSC* [17].)
(2) For a complete check of the SCB, the inputs of each gate must receive check tests when supplying sets of argument values to the inputs.
(3) The devices $F(X)$ and $G(X)$ must be checking (self-testing), which requires each fault to be manifested as a distortion of their output values on at least one set of argument values [25].

The check test is determined according to the structures of the devices in a CED circuit. The SCB involves XORs; for their canonical implementation, a complete check requires the use of all four combinations $\{00, 01, 10, 11\}$ [26]. In the noncanonical implementation case, a smaller number of test combinations may be taken, i.e., one of the following sets of combinations: $\{00, 01, 11\}$, $\{00, 01, 10\}$, $\{00, 10, 11\}$, and $\{01, 10, 11\}$ [27]. The set of test combinations necessary to check the checker in a CED circuit depends on its implementation. For example, for separable codes and an implementation in the form of an encoder and a comparator, it is required to generate the complete set of check vectors when supplying given sets of argument values to the inputs.

Application of BSC in CED circuit design implies computation control of the functions implemented at SCB outputs via a predetermined diagnostic attribute. For example, in the case of BSC, the self-duality of the functions [28] or the belonging of the generated codewords to preselected constant-weight codes [10, 11] can be controlled. Both of these diagnostic attributes were combined in [29].

When designing a CED circuit based on BSC, it is unnecessary to correct all signals from an object under diagnosis. In some cases, it suffices to convert only part of them. For example, in the case of a 2-out-of-4 constant-weight code, the object's outputs are divided into groups of four, and only two signals in each group are converted in the SCB [30]. However, converting all object's signals gives the designer greater flexibility in building self-checking devices, including the ability to choose the best implementation method. Let us demonstrate this below, using CED circuit design based on BSC and weighted sum codes as an example.

4. WEIGHTED SUM CODE IN THE RESIDUE RING OF A GIVEN MODULUS

Weighted sum codes represent a large class of uniform block codes that can be effectively used in the design of self-checking, controllable, and fault-tolerant automation and computing devices [8]. There are many variants to construct them under given redundancy constraints. The codewords of weighted sum codes are obtained by the following algorithm.

Algorithm 1 (the rules for forming codewords of weighted sum codes).

(1) Choose and fix m as the number of data symbols of the code. Order and combine them into the data vector $< f_m(X)f_{m-1}(X)\dots f_2(X)f_1(X) >$, where $f_i(X) \in \{0, 1\},\ i = \overline{1, m}$.
(2) Fix an array of weights $[w_m, w_{m-1}, \dots, w_2, w_1]$ assigned to the bits of the data vector, where $w_i \in \mathbb{N},\ i = \overline{1, m}$.
(3) Specify a modulus $M \in \mathbb{N},\ M \geqslant 2$.
(4) For each data vector, determine the smallest nonnegative residue for the sum of the weights of the significant bits:

$$W_M = W(\mathrm{mod}M) = \left(\sum_{i=1}^{m} w_i f_i(X)\right)(\mathrm{mod}M). \tag{3}$$

(5) Represent the resulting number in binary form and write it in $k = \lceil \log_2 M \rceil$ bits of the check vector.

(6) Concatenate the check vector to the data vector and write it in the lower bits of the codewords.

Hereinafter, we will designate weighted sum codes by $WS(m, k, M)$-codes, separately indicating the array of the weights $[w_m, w_{m-1}, \ldots, w_2, w_1]$ assigned to the bits of the data vector.[1]

An arbitrary modulus can be taken to construct a weighted sum code; meanwhile, special properties are observed for the codes with moduli $M \in \left\{2^1, 2^2, \ldots, 2^{\lceil \log_2(\sum_{i=1}^{m} w_i + 1)\rceil}\right\}$. For these codes, when considering the complete set of data vectors, all possible check vectors with k bits are generated at least once. (Their set has cardinality 2^k.) Therefore, the self-checking property of the checkers of such codes can be easily ensured.

Moreover, for definite values of the weights $[w_m, w_{m-1}, \ldots, w_2, w_1]$ and the moduli $M \in \left\{2^1, 2^2, \ldots, 2^{\lceil \log_2(\sum_{i=1}^{m} w_i + 1)\rceil}\right\}$, it is possible to achieve an important property from the standpoint of using $WS(m, k, M)$-codes in CED circuit design, i.e., the uniform distribution of the complete set of data vectors (of cardinality 2^m) among all check vectors with k bits from their complete set. With this property, it is easier to ensure the self-checking property of the CED circuit.

In CED circuit design, $WS(m, k, M)$-codes with various parameters can be used. The choice of code parameters is determined by the individual features of the object under diagnosis. If the detecting characteristics of the code allow covering the complete set of errors at the object's outputs, then a code with $n = m + k$ bits in codewords can be chosen. If complete error coverage is unachievable, separate codes are used to control subsets of outputs. This method of computation control in a CED circuit has other advantages over control of the complete set of outputs as well: it is much easier to ensure the self-checking property of individual control devices, and the checkers of the codes have simpler structures.

Below, we describe a CED circuit design method with BSC and $WS(m, k, M)$-codes that is based on correcting all signals from the object under diagnosis but uses a particular subset of codewords of the code instead of all codewords. This method makes it possible to fix the multiplicities of detectable errors at the object's outputs and adjust the structural redundancy and controllability indicators of CED circuits.

5. EXTRACTING A PARTICULAR SUBSET OF CODEWORDS OF A WEIGHTED SUM CODE FOR CED CIRCUIT DESIGN

To extract a particular subset of codewords of the $WS(m, k, M)$-code for CED circuit design, recall that all 2^k check vectors must be generated at least once. Otherwise, it will be impossible to ensure a complete check of the checker of the $WS(m, k, M)$-code. In view of the aforesaid, we propose the following algorithm for extracting a particular subset of codewords of the $WS(m, k, M)$-code for CED circuit design.

Algorithm 2 (the generalized algorithm for selecting a particular subset of codewords of the $WS(m, k, M)$-code).

(1) Fix the parameters of the $WS(m, k, M)$-code and the array of weights $[w_m, w_{m-1}, \ldots, w_2, w_1]$ assigned to data vector bits.

(2) Generate the complete set of codewords of the $WS(m, k, M)$-code of cardinality 2^m.

(3) Classify the codewords of the $WS(m, k, M)$-code into M subsets corresponding to the numbers W_M.

[1] More precisely, they can be called codes with summation of the weights of data vector bits in the residue ring of a given modulus.

(4) Form an M-partite graph (for codes with the uniform distribution of data vectors among check vectors, a Turán graph $T(2^m, M)$):
   (a) Assign to the vertices of each part the codewords corresponding to the numbers W_M (3). Then each part of the graph will contain the codewords with the same check vector.
   (b) Assign to the edges connecting the ith and jth vertices (v_i and v_j, respectively) the weights d_{ij} equal to the Hamming distance between the corresponding codewords.
(5) On the graph $T(2^m, M)$, select the maximal cliques (their size will be M).
(6) Determine a criterion for selecting a clique among the maximal ones.
(7) Select the cliques satisfying this criterion among the maximal cliques.
(8) Take an arbitrary clique among the remaining maximal ones.
(9) The codewords corresponding to the vertices in the selected maximal clique form a particular subset of codewords of the $WS(m, k, M)$-code for CED circuit design.

Let us demonstrate Algorithm 2 on an example of selecting a particular subset of codewords of the $WS(4, 2, 4)$-code with the array of weights $[w_4, w_3, w_2, w_1] = [1, 1, 2, 3]$; it can be used to build the "basic" CED circuit structure for groups of $n = 6$ outputs. For example, its analog with the conversion of some signals from the object under diagnosis, responsible for forming the data bits of the $WS(4, 2, 4)$-code, was considered in [31].

We generate all codewords of the $WS(4, 2, 4)$-code and classify them into groups corresponding to the numbers W_4 (Table 1). Next, we form a four-partite graph (in this case, a Turán graph $T(16, 4)$; see Fig. 2).

Table 1. Classification of codewords of the $WS(4, 2, 4)$-code

W_4			
0	1	2	3
g_2g_1			
00	01	10	11
Codewords			
0000 00	0011 01	1011 10	1010 11
0101 00	0100 01	0111 10	0110 11
1001 00	1000 01	0010 10	0001 11
1110 00	1101 01	1100 10	1111 11

In a Turán graph, all 2^m vertices are split into M parts of equal size $l = \frac{2^m}{M}$. Since $M \in \left\{2^1, 2^2, \ldots, 2^{\lceil \log_2(\sum_{i=1}^m w_i + 1)\rceil}\right\}$, the graph is regular ($2^m$ is divisible by M). Each vertex of the graph $T(2^m, M)$ has degree

$$\deg(v) = 2^m - \frac{2^m}{M} = 2^m\Big(1 - \frac{1}{M}\Big). \tag{4}$$

In the case under consideration, the graph $T(16, 4)$, each of the 16 vertices has degree $2^4\Big(1 - \frac{1}{4}\Big) = 12$.

The number of edges in a graph $T(2^m, M)$ for $WS(m, k, M)$-codes is given by

$$N_{\text{ver}} = \frac{(M-1)(2^m)^2}{2M} = \frac{M-1}{M}2^{2m-1}. \tag{5}$$

For the $WS(4, 2, 4)$-code, we have $\frac{4-1}{4}2^{2\cdot 4-1} = \frac{3}{4}2^7 = 96$.

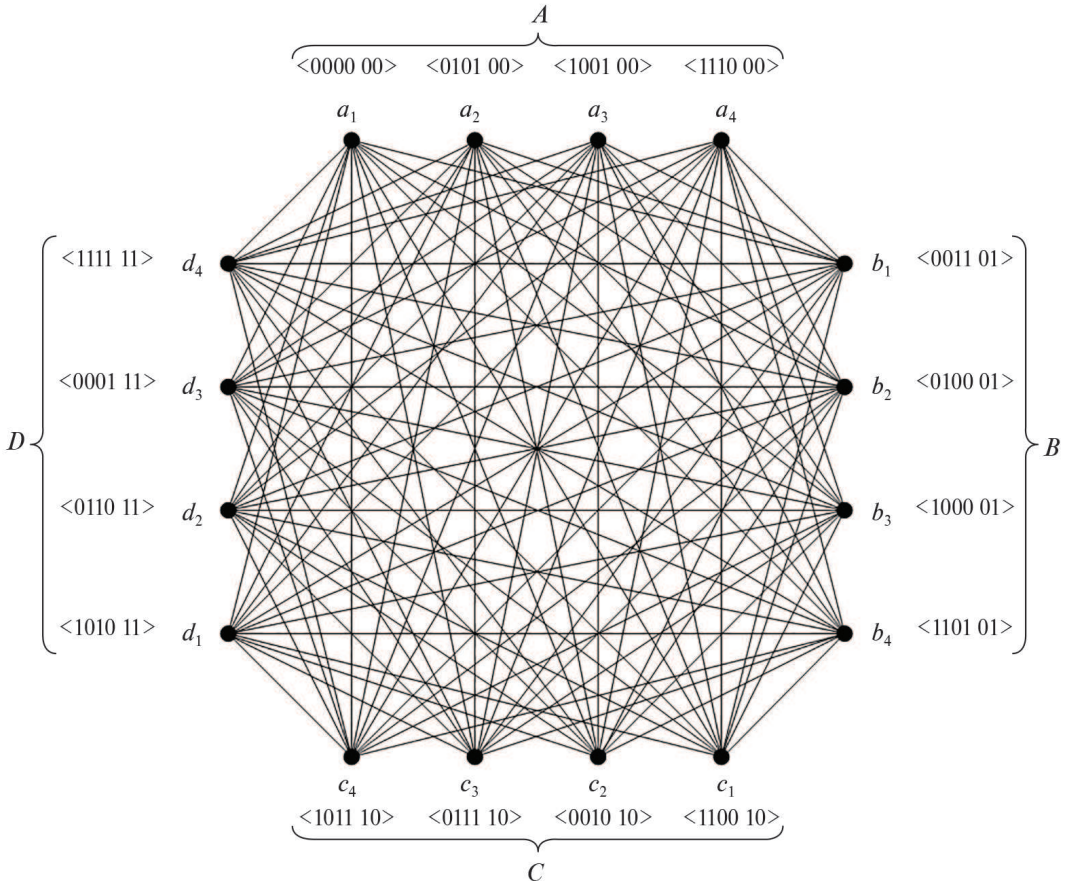

Fig. 2. The Turán graph $T(16,\ 4)$ for the $WS(4,\ 2,\ 4)$-code with the array of weights $[w_4, w_3, w_2, w_1] = [1,\ 1,\ 2,\ 3]$.

Next, it is necessary to extract all maximal cliques from the resulting graph $T(2^m, M)$. Their size equals M, and the number of edges is given by

$$R = \frac{M(M-1)}{2}. \tag{6}$$

For example, in the maximal cliques of the Turán graph built for the $WS(4, 2, 4)$-code, there are $\frac{4(4-1)}{2} = 6$ edges.

Each maximal clique corresponds to a particular subset of codewords of the $WS(m, k, M)$-code for CED circuit design. The number of maximal cliques in a graph $T(2^m, M)$ depends on the number of vertices in each part $(\frac{2^m}{M})$:

$$N_C = \Big(\frac{2^m}{M}\Big)^M. \tag{7}$$

For the $WS(4, 2, 4)$-code, we have $(\frac{2^4}{4})^4 = 2^8 = 256$.

Among the set of maximal cliques, it is necessary to select those with definite properties (satisfying a desired criterion).

The choice of an appropriate maximal clique will be associated with fixing certain values of the Hamming distance between the corresponding codewords. If the structure of the object under diagnosis (i.e., the configurations of gates and their connections to the device outputs) is unknown, then a reasonable criterion is the maximum total weight of the edges under the maximum shift of

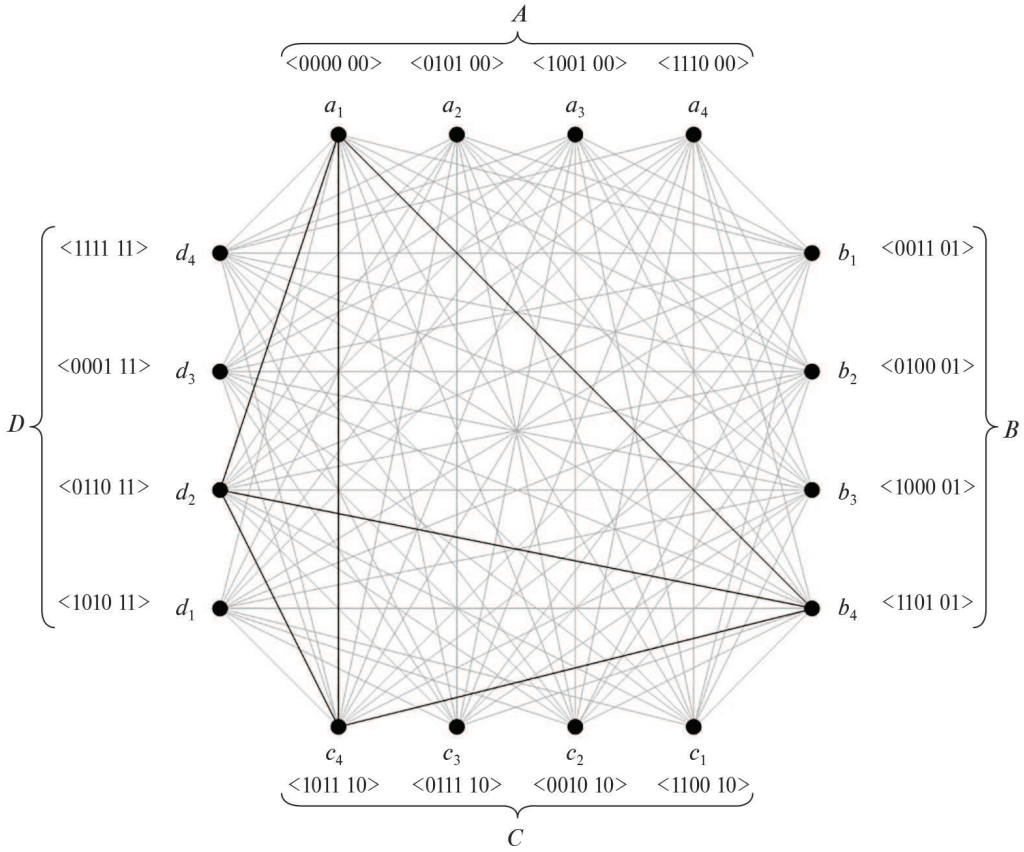

Fig. 3. Extracting a maximal clique in the Turán graph $T(16,\ 4)$ for the $WS(4, 2, 4)$-code with the array of weights $[w_4,\ w_3,\ w_2,\ w_1] = [1,\ 1,\ 2,\ 3]$.

the absolute values of the edge weights upward:

$$D_C = \sum_{v_i, v_j} d_{ij}, \quad d_{ij} \to \max, \tag{8}$$

where C is an expression containing all vertices of the clique (for large cliques, simply the clique number after ordering them), d_{ij} is the weight of the edge, and v_i and v_j are vertices belonging to this clique.

The maximum shift of absolute edge weights upward will be achieved if each weight of an edge in the maximal clique with total weight is as close as possible to

$$\gamma = \frac{D_C}{R} = \frac{2\sum_{v_i, v_j} d_{ij}}{M(M-1)}. \tag{9}$$

In the case of the $WS(4, 2, 4)$-code, formula (9) gives $\gamma = \frac{D_C}{6}$.

Note that the problem considered here (selecting a particular subset of codewords of the $WS(m, k, M)$-code for CED circuit design) is similar to the problem of selecting controllable probabilistic tests addressed, e.g., in [32, 33].

There are 256 maximal cliques in a graph $T(2^m, M)$. After an exhaustive search of all these, we identified a unique maximal clique for which all edge weights equal 4, and the total weight of the clique is 24. ($D_{a_1,b_4,c_4,d_2} = 24$ and $\gamma = \frac{24}{6} = 4$.) The clique includes vertices a_1, b_4, c_4, and d_2 (Fig. 3). The particular subset of codewords of the $WS(4, 2, 4)$-code corresponding to this clique is $\{< 0000\ 00 >, < 1101\ 01 >, < 1011\ 10 >, < 0110\ 11 >\}$. They will be used below to design of a CED circuit based on BSC with the $WS(4, 2, 4)$-code.

There exist other maximal cliques with the value $D_C = 24$. For example, the same number is provided by extracting a maximal clique with vertices a_1, b_1, c_1, and d_4. However, for the codewords corresponding to this clique, four pairs will have code distances equal to 3, and two pairs equal to 6. The above criterion allows discarding all cases except those with the maximum shift of weights upward.

When selecting a maximal clique on a Turán graph for codewords of the $WS(m, k, M)$-code, the number of maximal cliques grows astronomically with m. Therefore, in practice, one can establish either particular subsets of codewords of $WS(m, k, M)$-codes with various properties (one-time) or a criterion for selecting a maximal clique corresponding to the desired conditions (not necessarily the maximum possible Hamming distance between codewords). For example, such a condition is to select a maximal clique with all edge weights $d_{ij} \geqslant 3$.

Some $WS(m, k, M)$-codes can be used in CED circuit design based on two diagnostic attributes, i.e., the belonging of codewords to a preselected code and the self-duality of functions [34]. Not all $WS(m, k, M)$-codes possess this property. The $WS(4, 2, 4)$-code with the array of weights $[w_4, w_3, w_2, w_1] = [1, 1, 2, 3]$ (see above) is a representative of the class of codes with check symbols described by self-dual Boolean functions. Such codes have the following feature: on the complete set of their codewords, it is possible to select 2^{m-1} pairs with words orthogonal over all bits. Direct analysis of Table 1 shows that the $WS(4, 2, 4)$-code possesses this property. When building a CED circuit based on BSC with computation control via two diagnostic attributes, this fact can be utilized to establish a criterion for extracting a particular subset of codewords. For example, consider the Turán graph for the $WS(4, 2, 4)$-code (Fig. 2) and select two different vertex pairs of the Turán graph $(a_{i_1}, b_{i_2}) \,\&\, (c_{i_3}, d_{i_3}) \vee (a_{i_1}, c_{i_3}) \,\&\, (b_{i_2}, d_{i_3}) \vee (a_{i_1}, d_{i_4}) \,\&\, (b_{i_2}, c_{i_3})$, where i_1, i_2, i_3, i_4 denote the vertex numbers in the corresponding parts of the graph $(1 - A, 2 - B, 3 - C, 4 - D)$, with the maximum edge weight $\nu_{ij} = 6$. Then, the selected codewords will allow computation control via two diagnostic attributes under definite conditions for generating codewords $< h_n(X)h_{n-1}(X) \dots h_2(X)h_1(X) >$. (On orthogonal input combinations over all arguments, it is necessary to fix codewords also orthogonal over all arguments.) An example of a maximal clique satisfying this criterion is the clique with vertices a_1, b_1, c_1, and d_4. The corresponding codewords are $\{< 0000\ 00 >, < 0011\ 01 >, < 1100\ 10 >, < 1111\ 11 >\}$.

6. BLOCK DESIGN FEATURES IN A CED CIRCUIT

In the CED circuit structure (Fig. 1), the SCB has a standard implementation of n XORs. The block $G(X)$ is essentially an encoder of the $WS(m, k, M)$-code, intended to compute the bits of its codewords when supplying sets of argument values to the inputs; and TSC is a device for recognizing the codewords of this code. In the presented implementation, one could use the standard structure of a checker of any separable code in the form of a cascade connection of the encoder $G(F)$ and the comparator $kTRC1$: the former generates a check vector from the values of the data vector, and the latter compares the bits of the check vector obtained directly from SCB outputs and the check vector generated by the encoder $G(F)$, converting k two-rail signals into one. The self-checking comparator is implemented in two-rail logic from standard modules for compressing two-rail signals (TRC, two-rail checkers) [35]; therefore, in this checker implementation, it is necessary to invert the bits of the check vector generated at SCB outputs. One could immediately design the block $G(X)$ considering the need to obtain not the check vector for the $WS(m, k, M)$-code but the inverted vector. This would be sufficient for the correct operation of the CED circuit. However, in this case, it becomes impossible to take into account a particular subset of codewords of the $WS(m, k, M)$-code instead of its complete set. For instance, any distortion converting a codeword of the $WS(m, k, M)$-code into a codeword of the same code, even when using some particular subset of codewords of this code at the CED circuit design stage, will not be detected by the checker: a two-rail signal will

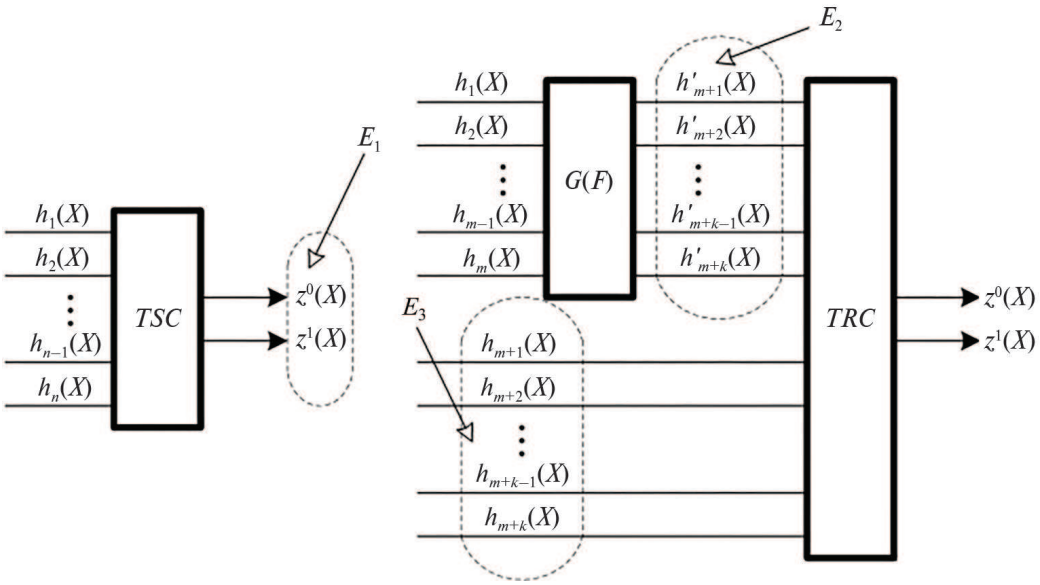

Fig. 4. Areas in the circuits of checkers of the $WS(4, 2, 4)$-code suitable for structural modifications.

be generated at its outputs. Therefore, to consider the error detection properties in the codewords of a particular subset of codewords of the $WS(m, k, M)$-code, the checker must be modified to "respond" only to the codewords preselected by the CED circuit designer.

Figure 4 shows the structures of checkers of the $WS(m, k, M)$-code and possible circuit nodes suitable for modifying signals on the checker lines, i.e., the areas E_1, E_2, and E_3. The checker can be implemented as a codeword detector: when codewords from a particular subset are received at its inputs, a two-rail signal is generated at the checker outputs; otherwise, a non-two-rail signal is generated. This modification area is E_1. The checker can be implemented as a two-stage structure consisting of an encoder and a comparator. In this case, there are two possible modification areas: modifying signals at the output of the encoders $G(F)$ and $G(X)$ (the areas E_2 and E_3, respectively).

Let us demonstrate possible structural modifications of the checker of the $WS(m, k, M)$-code on an example of the $WS(4, 2, 4)$-code with the array of weights $[w_4, w_3, w_2, w_1] = [1, 1, 2, 3]$ and the particular subset of codewords $\{< 0000\ 00 >, < 1101\ 01 >, < 1011\ 10 >, < 0110\ 11 >\}$.

Table 2 presents the options for fixing signals in the highlighted areas for modifying the control devices in the CED circuit.

Consider the option of designing the checker as a detector; then it is implemented in the form of a converter so that for each codeword belonging to the $WS(4, 2, 4)$-code, one of the two vectors from column E_1 is generated. Note that in the case of computation errors, the device $F(X)$ or the CED circuit blocks can generate a codeword not belonging to the $WS(4, 2, 4)$-code; hence, it is required to generate the values $< 00 >$ or $< 11 >$ for all codewords not belonging to this code. Accordingly, there exist $2^{2^6} = 2^{64}$ variants to specify the required values and build such a detector.

We have the following options for building a modified encoder of the $WS(4, 2, 4)$-code. When implementing the encoder $G(F)$ or the encoder $G(X)$, it is necessary to exclude the generation of a correct check vector when a codeword not belonging to the $WS(4, 2, 4)$-code arrives at the inputs. In the case under consideration, this can be done in three variants for each codeword unused. For codewords not belonging to the $WS(4, 2, 4)$-code, no special complement is required: if such a word appears, a mismatch between the data and check vectors will be recorded at SCB outputs. There are 3^{12} variants to specify the required values and build the block $G(F)$ and, accordingly, the block $G(X)$.

Table 2. Options for fixing signals under control device modifications in the CED circuit

Codewords of the code			Signals in modification areas	
Data vector	Check vector	Belonging to the particular subset of codewords	E_1	E_2, E_3
0000	00	+	01 / 10	00
0001	11	–	00 / 11	00 / 01 / 10
0010	10	–	00 / 11	00 / 01 / 11
0011	01	–	00 / 11	00 / 10 / 11
0100	01	–	00 / 11	00 / 10 / 11
0101	00	–	00 / 11	01 / 10 / 11
0110	11	+	01 / 10	11
0111	10	–	00 / 11	00 / 01 / 11
1000	01	–	00 / 11	00 / 10 / 11
1001	00	–	00 / 11	01 / 10 / 11
1010	11	–	00 / 11	00 / 01 / 10
1011	10	+	01 / 10	10
1100	10	–	00 / 11	00 / 01 / 11
1101	01	+	01 / 10	01
1110	00	–	00 / 11	01 / 10 / 11
1111	11	–	00 / 11	00 / 01 / 10

The trivial issue of the structural design of modified encoders, checkers, and detectors of $WS(m, k, M)$-codes (in particular, the $WS(4, 2, 4)$-code) goes beyond the scope of this paper.

7. PRINCIPLES FOR SPECIFYING VALUES AT SCB OUTPUTS

There are numerous variants to form codewords at SCB outputs. Even for a preselected particular subset of codewords, a large number of variants can be used to specify the values of the functions $h_1(X), \ldots, h_n(X)$ and, accordingly, $g_1(X), \ldots, g_n(X)$.

Generally, it is required to fix appropriate values of the functions $h_1(X), \ldots, h_n(X)$ on each of the 2^t sets of argument values in order to form codewords from a preselected particular subset of codewords of the $WS(m, k, M)$-code. In doing so, the above conditions for ensuring the self-checking property of the CED circuit blocks should be considered. Clearly, each check vector of the $WS(m, k, M)$-code can be generated at least once under the condition $t \geqslant M$; otherwise, this is impossible. If the condition holds, the BSC functions can be obtained by sequentially selecting codewords from the particular subset of codewords of the $WS(m, k, M)$-code for each row of the truth table that describes the object under diagnosis and the outputs of the CED circuit blocks. The particular subset of codewords has cardinality M, and on each of the 2^t rows, there are M options for obtaining codewords. However, the upper bound is M^{2^t} (while neglecting the generation of each unique codeword once). With the latter factor taken into account, on M rows we have to generate at least each codeword from the selected particular subset of codewords of the $WS(m, k, M)$-code, and on the remaining $2^t - M$ rows proceed arbitrarily. In total, there are $M^{2^t - M}$ variants to specify the values of the functions $h_1(X), \ldots, h_n(X)$ (accordingly, the same number of variants to design the CED circuit with a fixed particular subset of codewords). For example, for the $WS(4, 2, 4)$-code with $t = 4$, we have $4^{2^4 - 4} = 4^{12} = 16\,777\,216$ options for CED circuit design. Among these options, some may be less efficient, e.g., those not ensuring the self-checking property of the CED circuit

blocks or leading to high structural redundancy of the device (as compared to the redundancy of a self-checking implementation of the device by duplication or another method).

From the standpoint of ensuring the testability of CED circuit gates during the operation of a self-checking device, a method of interest is to specify appropriate values of the functions $h_1(X), \ldots, h_n(X)$ so that the probabilities of generating each test combination for the checker are close (if not equal). This can be achieved by uniformly distributing the codewords from a particular subset of codewords of the $WS(m, k, M)$-code among all 2^t input combinations. However, the uniform distribution of codewords does not guarantee the possibility of generating tests for SCB gates, but one can always return to the step of determining the values of the BSC functions and change their fixation method.

Algorithm 3 (the rules for designing CED circuits based on BSC with a fixed particular subset of codewords of the $WS(m, k, M)$-code).

(1) Order the codewords from a fixed particular subset of codewords of the $WS(m, k, M)$-code and number them by $q \in \{1, \ldots, 2^M\}$.
(2) Form a truth table with 2^t rows to describe the logic of the object under diagnosis (as noted above, the method works correctly if $t \geqslant M$.)
(3) Determine the so-called *repetition coefficient* of codewords:

$$\delta = \frac{2^t}{M}. \tag{10}$$

(4) On the rows of the truth table with decimal numbers (they are assigned decimals $0, \ldots, 2^{t-1}$, corresponding to the binary numbers written in the set of argument values) from the range $(q-1)\delta \ldots q\delta - 1$, fix the signals at SCB outputs so that the codeword with number q is generated.
(5) Obtain the values of the BSC functions using (2).
(6) Verify the condition for generating tests for all SCB gates. If a complete test cannot be generated, change the method for fixing the signals.
(7) Design the block $G(X)$.
(8) Select a method for modifying the control part of the CED circuit and modify one of the devices in the circuit.

Let us emphasize several important aspects in the operation of Algorithm 3. First, it is unnecessary to control the formation of the complete set of test combinations for the checker: all check vectors will be generated automatically. Second, a complete test for each SCB gate will not be formed in some cases (see Step 6 of Algorithm 3). If this happens, the variant for determining the values should be changed. One could initially focus on the row-by-row analysis of the tables describing the operation of devices, considering the generation of test combinations for gates, as done in [36]; in this case, however, the algorithm will run longer. (This is the main vulnerability of the algorithm.)

We demonstrate the operation of Algorithm 3 to obtain a description of a CED circuit on each line using the $WS(4, 2, 4)$-code with the array of weights $[w_4, w_3, w_2, w_1] = [1, 1, 2, 3]$ and a particular subset of its codewords $\{< 0000\ 00 >, < 1101\ 01 >, < 1011\ 10 >, < 0110\ 11 >\}$ for control of the device specified by Table 3.

In the CED circuit (see Fig. 1), the SCB has a standard implementation. It is required to design $G(X)$ and *TSC*.

Following the steps of Algorithm 3, we obtain the values of the functions describing the device $G(X)$. After ordering, the codewords are assigned the numbers: 1—$< 0000\ 00 >$, 2—$< 1101\ 01 >$, 3—$< 1011\ 10 >$, and 4—$< 0110\ 11 >$. By formula (10), the repetition coefficient of codewords from the particular subset is $\delta = \frac{2^4}{4} = 4$. Next, on rows with numbers $0, \ldots, 3$, we specify the

Table 3. The operation of a device with four outputs and the CED circuit for it

nos.	The sets of argument values				The values at the outputs of the device $F(X)$							The values at the outputs of the block $G(X)$							The values at the outputs of the SCB							Test combinations for SCB gates						
	x_4	x_3	x_2	x_1	f_6	f_5	f_4	f_3	f_2	f_1	g_6	g_5	g_4	g_3	g_2	g_1	h_6	h_5	h_4	h_3	h_2	h_1	XOR_6	XOR_5	XOR_4	XOR_3	XOR_2	XOR_1
0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	00	11	11	11	00	00
1	0	0	0	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	0	0	0	0	00	11	00	00	11	11
2	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	00	00	11	11	00	11
3	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	11	00	00	00	00	00
4	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	1	0	1	0	1	10	10	00	01	00	10
5	0	1	0	1	0	0	1	0	0	1	1	1	1	1	0	0	1	1	0	1	0	1	01	01	11	01	00	10
6	0	1	1	0	1	0	0	1	0	0	0	1	0	0	0	1	1	1	0	1	0	1	10	01	00	10	00	01
7	0	1	1	1	0	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	01	10	11	10	11	10
8	1	0	0	0	1	0	1	0	0	0	0	0	0	1	1	0	1	0	1	1	1	0	10	00	10	01	01	00
9	1	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	1	0	1	1	1	0	01	00	10	01	10	00
10	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	10	00	01	10	10	00
11	1	0	1	1	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	1	1	0	10	00	01	10	01	00
12	1	1	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	1	1	0	1	1	11	10	01	00	01	01
13	1	1	0	1	0	1	1	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	00	10	10	00	01	10
14	1	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1	1	0	1	1	00	01	10	11	01	01
15	1	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1	0	1	1	00	01	01	11	01	01

signals at SCB outputs so that codeword 1 is formed; on rows with numbers $4, \ldots, 7$, codeword 2; on rows with numbers $8, \ldots, 11$, codeword 3; finally, on rows with numbers $12, \ldots, 15$, codeword 4. Using (2), we determine the values of the BSC functions; see the results in Table 3, where the last six columns present the combinations generated at the inputs of the gates. Direct analysis of these columns indicates that a test combination is formed for each gate.

Next, we select a method for modifying the control part of the CED circuit and design the self-checking device. Here, the issues of structural design and efficiency indicators of the resulting CED circuit are omitted: the presentation is intended to show only the fairly simple and understandable principles of implementing a self-checking device.

According to experimental studies on a variety of examples, in a large number of cases, Algorithm 3 yields less redundant self-checking combinational discrete devices compared to duplication. The effect is higher for more complex initial functions computed by the block $F(X)$. When designing self-checking finite state machines (FSMs) with CED circuits for logic and output converters, this effect only increases since duplication of all Boolean blocks of FSMs is not required.

8. CONCLUSIONS

This paper has described the features of the authors' method for designing self-checking devices based on BSC in a CED circuit with a particular subset of codewords of the $WS(m, k, M)$-code and conversion of all signals from an object under diagnosis.

The advantages of this CED circuit design method are obvious. First, by choosing a particular subset of codewords of the $WS(m, k, M)$-code, it is possible to consider the characteristics of errors occurring at the object's outputs. Second, there are numerous variants to specify the values of the functions formed at SCB outputs, which provides "flexibility" and the ability to choose the CED circuit structure during its design. Third, it is easy to modify the control part of the CED circuit so that it detects any errors except those distorting codewords from the selected particular subset of codewords of the $WS(m, k, M)$-code into each other. In fact, this method is closely related to those involving inseparable codes in CED circuit design with BSC [10, 11, 15, 16, 30], and the above methods for modifying the control circuits are based on modifying the $WS(m, k, M)$-code.

Note the following drawbacks of the CED circuit design method. The most serious one, in the authors' opinion, is the necessity to ensure the self-checking property of SCB gates and the checker of the $WS(m, k, M)$-code. This cannot be done for all devices. For example, all SCB gates are checked only if each of the functions computed by the object under diagnosis takes value 1 (or 0) on at least two sets of argument values [31]. With a small number of the object's inputs, this is not always feasible. The second drawback is that, to ensure a complete check, a definite subset of argument values has to be supplied during the operation of the self-checking device. This cannot be done, e.g., in critical systems where input actions change infrequently [37], and a combination of test and functional diagnosis methods is then required [38]. Concerning the third drawback, the method itself requires the selection of a particular subset of codewords of the $WS(m, k, M)$-code, which (albeit being performed once) is difficult for large values of the code parameters m, k, M. In addition, there are numerous variants to specify the values of the functions at SCB outputs on each set of argument values; despite the possibility to choose a single variant for fixing their values, in many practical cases, an optimal one cannot be chosen due to the complexity (or even infeasibility) of an exhaustive enumeration of all variants.

From the authors' point of view, the above method for designing self-checking devices is of interest for implementation on modern programmable logic devices.

REFERENCES

1. Efanov, D.V., *Metody sinteza samoproveryaemykh diskretnykh ustroistv* (Design Methods for Self-Checking Discrete Devices), Moscow: LENAND, 2025.
2. Sagalovich, Yu.L., *Algebra, kody, diagnostika* (Algebra, Codes, Diagnosis), Moscow: Institute for Information Transmission Problems, Russian Academy of Sciences, 1993.
3. Fujiwara, E., *Code Design for Dependable Systems: Theory and Practical Applications*, John Wiley & Sons, 2006.
4. Goessel, M., Ocheretny, V., Sogomonyan, E., and Marienfeld, D., *New Methods of Concurrent Checking*, 1st ed., Dordrecht: Springer Science+Business Media B.V., 2008.
5. Drozd, A.V., Kharchenko, V.S., Antoshchuk, S.G., et al., *Rabochee diagnostirovanie bezopasnykh informatsionno-upravlyayushchikh sistem* (Operational Diagnosis of Safe Information and Control Systems), Khar'kov: Zhukovskii National Aerospace University, 2012.
6. Mikoni, S.V., Sokolov, B.V., and Yusupov, R.M., *Kvalimetriya modelei i polimodel'nykh kompleksov* (Qualimetry of Models and Polymodel Complexes), Moscow: Russian Academy of Sciences, 2018.
7. 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.
8. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., *Kody s summirovaniem dlya sistem tekhnicheskogo diagnostirovaniya. Tom 2: Vzveshennye kody s summirovaniem* (Sum Codes for Technical Diagnosis Systems. Vol. 2: Weight-Based Codes), Moscow: Nauka, 2021.
9. Das, D., Touba, N.A., Seuring, M., and Gossel, M., Low Cost Concurrent Error Detection Based on Modulo Weight-Based Codes, *Proceedings of the IEEE 6th International On-Line Testing Workshop (IOLTW)*, Spain, Palma de Mallorca, July 3–5, 2000, pp. 171–176. https://doi.org/10.1109/OLT.2000.856633
10. Gessel, M., Morozov, A.V., Sapozhnikov, V.V., et al., Logic Complement, a New Method of Checking the Combinational Circuits, *Autom. Remote Control*, 2003, vol. 64, no. 1, pp. 153–161.
11. Goessel, M., Morozov, A.V., Sapozhnikov, V.V., et al., Checking Combinational Circuits by the Method of Logic Complement, *Autom. Remote Control*, 2005, vol. 66, no. 8, pp. 1336–1346.
12. Sogomonyan, E.S. and Slabakov, E.V., *Samoproveryaemye ustroistva i otkazoustoichivye sistemy* (Self-Checking Devices and Fault-Tolerant Systems), Moscow: Radio i Svyaz', 1989.
13. Mitra, S. and McCluskey, E.J., Which Concurrent Error Detection Scheme to Choose?, *Proceedings of the International Test Conference*, Atlantic City, NJ, October 3–5, 2000, pp. 985–994. https://doi.org/10.1109/TEST.2000.894311
14. Efanov, D.V., The Equal-Length Redundant Code Development for the Self-Checking Combinational Devices Synthesis Based on Data on Their Structures, *Electronic Modeling*, 2022, vol. 44, no. 1, pp. 43–52. https://doi.org/10.15407/emodel.44.01.043
15. Sapozhnikov, V., Sapozhnikov, Vl., and Efanov, D., Concurrent Error Detection of Combinational Circuits by the Method of Boolean Complement on the Base of "2-out-of-4" Code, *Proceedings of 14th IEEE East-West Design & Test Symposium (EWDTS'2016)*, Yerevan, Armenia, October 14–17, 2016, pp. 126–133. https://doi.org/10.1109/EWDTS.2016.7807677
16. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., Formation of Totally Self-Checking Structures of Concurrent Error Detection Systems with Use of Constant-Weight Code "1-Out-Of-3", *Electronic Modeling*, 2016, vol. 38, no. 6, pp. 25–43.
17. Sapozhnikov, V.V. and Sapozhnikov, Vl.V., *Samoproveryaemye diskretnye ustroistva* (Self-Checking Discrete Devices), St. Petersburg: Energoatomizdat, 1992.
18. Sogomonyan, E.S. and Gossel, 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. https://doi.org/10.1007/BF00971975

19. Busaba, F.Y. and Lala, P.K., Self-Checking Combinational Circuit Design for Single and Unidirectional Multibit Errors, *Journal of Electronic Testing: Theory and Applications*, 1994, vol. 5, no. 5, pp. 19–28. https://doi.org/10.1007/BF00971960

20. Morosow, A., Saposhnikov, V.V., Saposhnikov, Vl.V., and Goessel, M., Self-Checking Combinational Circuits with Unidirectionally Independent Outputs, *VLSI Design*, 1998, vol. 5, no. 4, pp. 333–345. https://doi.org/10.1155/1998/20389

21. Efanov, D.V., Sapozhnikov, V.V., and Sapozhnikov, V.V., Conditions for Detecting a Logical Element Fault in a Combination Device under Concurrent Checking Based on Berger's Code, *Autom. Remote Control*, 2017, vol. 78, no. 5, pp. 891–901.

22. Efanov, D.V., Sapozhnikov, V.V., and Sapozhnikov, Vl.V., Organization of a Fully Self-Checking Structure of a Combinational Device Based on Searching for Groups of Symmetrically Independent Outputs, *Automatic Control and Computer Sciences*, 2020, vol. 54, no. 4. pp. 279–290. https://doi.org/10.3103/S0146411620040045

23. Aksenova, G.P. and Sogomonyan, E.S., Design of Self-Checking Built-In Check Circuits for Automata with Memory, *Autom. Remote Control*, 1975, vol. 36, no. 7, pp. 1169–1177.

24. 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. https://doi.org/10.17586/0021-3454-2023-66-5-355-372

25. 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.

26. Aksenova, G.P., Necessary and Sufficient Conditions for the Synthesis of Completely Testable Modulo 2 Convolution Circuits, *Autom. Remote Control*, 1979, vol. 40, no. 9, pp. 1362–1369.

27. Yelina, Y.I. and Efanov, D.V., Weight-Based Bose–Lin Codes in Concurrent Error-Detection Circuit Based on Boolean Signal Correction, *Journal of Computer and Systems Sciences International*, 2025, vol. 64, no. 1, pp. 17–35.

28. Saposhnikov, Vl.V., Dmitriev, A., Goessel, M., and Saposhnikov, V.V., Self-Dual Parity Checking — A New Method for On-line Testing, *Proceedings of the 14th IEEE VLSI Test Symposium*, USA, Princeton, 1996, pp. 162–168. https://doi.org/10.1109/VTEST.1996.510852

29. Efanov, D.V. and Pivovarov, D.V., The Hybrid Structure of a Self-Dual Built-In Control Circuit for Combinational Devices with Pre-Compression of Signals and Checking of Calculations by Two Diagnostic Parameters, *Proceedings of the 19th IEEE East-West Design and Test Symposium (EWDTS'2021)*, Batumi, Georgia, September 10–13, 2021, pp. 200–206. https://doi.org/10.1109/EWDTS52692.2021.9581019

30. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Efanov, D.V., Design of Self-Checking Concurrent Error Detection Systems Based on "2-out-of-4" Constant-Weight Code, *Probl. Upr.*, 2017, no. 1, pp. 57–64.

31. 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. http://doi.org/10.25728/cs.2024.4.3

32. Yarmolik, S.V. and Yarmolik, V.N., The Synthesis of Probability Tests with a Small Number of Kits, *Automatic Control and Computer Sciences*, 2011, vol. 45, no. 3, pp. 133–141. https://doi.org/10.3103/S0146411611030072

33. Yarmolik, V.N., Petrovskaya, V.V., and Mrozek, I., A Measure of Differences for Test Sets in Reducing Controllable Random Tests, *Informatics*, 2022, vol. 19, no. 4, pp. 7–26. https://doi.org/10.37661/1816-0301-2022-19-4-7-26

34. Efanov, D.V. and Pivovarov, D.V., Design of Self-Checking Discrete Devices Based on Polynomial Codes with Computation Control via Several Diagnostic Attributes, *Autom. Remote Control*, 2025, vol. 86, no. 5, pp. 402–416.

35. Lala, P.K., *Self-Checking and Fault-Tolerant Digital Design*, San Francisco: Morgan Kaufmann Publishers, 2001.

36. Efanov, D.V., Synthesis of Self-Checking Combinational Devices Based on the Boolean Signal Correction Method Using Bose–Lin Codes, *Information Technologies*, 2023, vol. 29, no. 10, pp. 503–511. https://doi.org/10.17587/it.29.503-511

37. Drozd, A., Kharchenko, V., Antoshchuk, S., et. al., Checkability of the Digital Components in Safety-Critical Systems: Problems and Solutions, *Proceedings of the 9th IEEE East-West Design and Test Symposium (EWDTS'2011)*, Sevastopol, Ukraine, 2011, pp. 411–416. https://doi.org/10.1109/EWDTS.2011.6116606

38. Litikov, I.P. and Sogomonyan, E.S., Test and Functional Diagnosis of Digital Devices and Systems, *Autom. Remote Control*, 1985, vol. 46, no. 3, pp. 375–384.

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