# COMPOSITIONS OF TWO CONSTANT-WEIGHT CODES WITH ORTHOGONAL COMBINATIONS OVER ALL BITS FOR SELF-CHECKING DISCRETE DEVICE DESIGN

D. V. Efanov

Peter the Great Saint Petersburg Polytechnic University, St. Petersburg, Russia
Russian University of Transport, Moscow, Russia

✉ TrES-4b@yandex.ru

**Abstract.** This paper proposes using compositions of two constant-weight codes with orthogonal combinations over all bits in the design of controllable and self-checking discrete devices. With such codes, computation control at the outputs of discrete devices can be implemented via the attribute of belonging of the codewords to a given constant-weight code and, moreover, via the attribute of belonging of each function describing the codeword bits to the class of self-dual Boolean functions. It is shown how to construct noninterference codes based on the composition of two constant-weight codes with orthogonal combinations over all bits. Explicit formulas are derived to determine the number of errors undetectable by compositions of constant-weight codes by their types (by the number of erroneous 0s and 1s in codewords) and multiplicities. The properties of the codes under consideration are briefly described. The structure of concurrent error-detection circuits is presented for discrete devices based on the composition of two constant-weight codes with orthogonal combinations over all bits and computation control via two diagnostic attributes. The use of such compositions can be effective in building highly reliable discrete devices on various components.

**Keywords**: controllable and self-checking devices; computation control via two diagnostic attributes; compositions of constant-weight codes; error detection at discrete device outputs.

## INTRODUCTION

Constant-weight codes, also called “*r*-out-of-*n*” or *r/n* codes, are constructed by selecting, from a set of codewords with *n* bits (with a codeword length of *n*), those having the same weight *r*. The number of codewords in an *r/n* code is determined by the binomial coefficient $C_n^r$. These codes were described in [1] and found wide application in data processing and transmission [2, 3] as well as in the design of controllable and self-checking discrete devices [4–7]. The design theory of detectors and checkers of constant-weight codes is rather deeply developed [8–12]; special properties and characteristics of these codes were investigated in several works, e.g., in the paper [13]. Note also that constant-weight codes are closely related to the theory of combinatorial block design and Steiner systems [14, 15].

We focus the reader’s attention on the application of constant-weight codes to build devices with fault detection [16, 17]. When designing such devices using constant-weight codes, the following advantages are utilized. First of all, since the codewords of these codes have the same weight, they detect any errors in the codewords except for multidirectional errors with even multiplicity containing the distortion group $\{0\to 1, 1\to 0\}$ (the so-called symmetrical errors). Constant-weight codes detect any non-symmetrical errors classified as unidirectional (associated with distortions of exclusively 0s or exclusively 1s) and asymmetrical (containing an unequal number of distortions of 0s and 1s). This feature is used in the design of self-checking devices by searching for groups of unidirectional-independent (UI) outputs [18, 19] or groups of unidirectional/asymmetrical-independent (UAI) outputs [20, 21] in a device and organizing computation control at these outputs by means of constant-weight codes. An alternative is to modify the structure of a controlled device into one with a single group of UI outputs or UAI outputs [22]. Another fruitful property is that constant-weight codes do not detect a small number of errors with even multiplicity, which also contributes to covering a large number of real distortions at the outputs of controlled devices. The third important advantage of constant-weight codes over other codes is the simplicity of control equipment (checkers, detectors, and check logic blocks) and clear principles of ensuring self-checking [9].

Different approaches can be applied to build self-checking discrete devices using constant-weight codes. It is interesting to control computations using not only constant-weight codes but also the properties of Boolean functions describing the bits of their codewords, namely, the properties of self-dual Boolean functions. Such an organization of computation control significantly improves controllability indicators in the part of observability [23]. When building discrete devices with computation control via two diagnostic attributes, only $\frac{n}{2}/n$ codes can be applied [24]. In this paper, the idea is to design self-checking discrete devices using compositions of two constant-weight codes with orthogonal combinations over all bits, which significantly extends the design theory of self-checking discrete devices.

## 1. COMPOSITIONS OF TWO CONSTANT-WEIGHT CODES WITH ORTHOGONAL COMBINATIONS OVER ALL BITS

Among all constant-weight codes, it is possible to separate those with the following interesting property: the set of their codewords contains exclusively the pairs of orthogonal codewords over all bits. Such codes exist only for even *n*, and there is a single code of this type with the value $r = \frac{n}{2}$ for each even *n*. The whole variety of constant-weight codes can be illustrated on Pascal's triangle, where each number characterizes the cardinality of the codeword set of *r/n* codes; see a fragment of Pascal's triangle in Fig. 1. That is, Fig. 1a shows the triangle with highlighted numbers corresponding to the cardinalities of the codeword sets of constant-weight codes with the above property. For example, the constant-weight 1/2 code contains the two combinations {01, 10} and is used in information coding for error detection, e.g., in the spatial two-rail representation of signals [25]. The constant-weight 2/4 code contains the six combinations {0011, 0101, 0110, 1001, 1010, 1100} and is used in organizing concurrent error-detection (CED) circuits for the combinational components of discrete devices [26].

In [24], it was proposed to use (constant-weight) $\frac{n}{2}/n$ codes in the design of self-checking discrete devices with computation control via two diagnostic attributes, namely, the belonging of codewords to a given constant-weight code and the belonging of each function describing the bits of codewords to the class of self-dual Boolean functions. For this purpose, the following property must be provided in the design process of a CED circuit: on the sets of argument values orthogonal over all bits, it is required to form, in the CED circuit, orthogonal codewords over all bits for constant-weight codes.

The studies of the applicability of constant-weight codes to computation control at discrete device outputs via two diagnostic attributes have led to the task of determining the applicability of groups of other constant-weight codes to CED circuit organization.

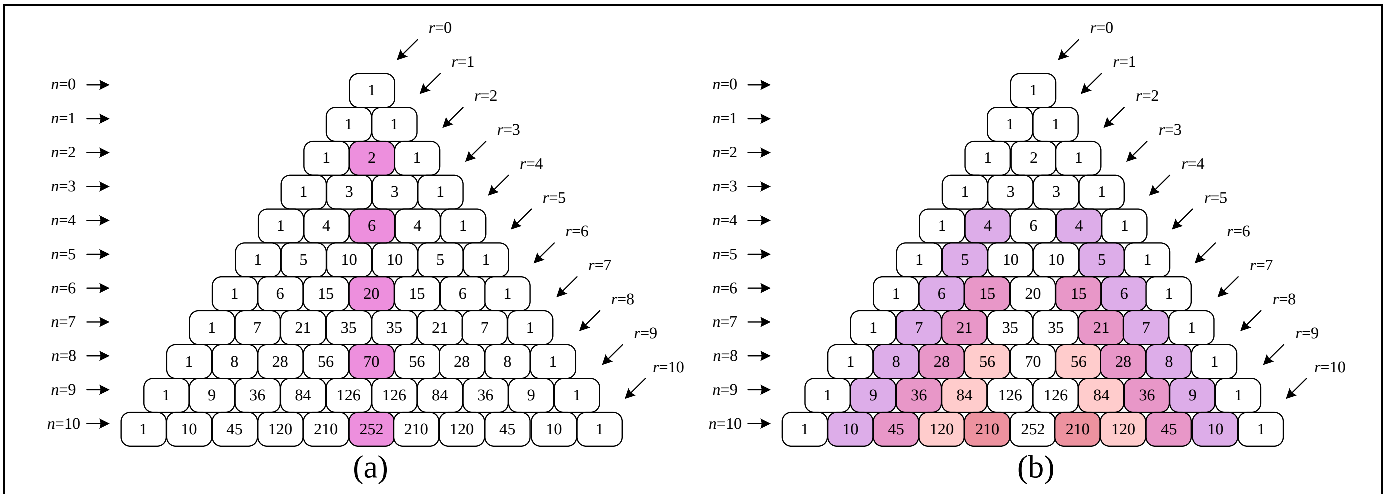

**Fig. 1. Ways to select a) one and b) two constant-weight codes with orthogonal codewords over all bits on Pascal's triangle.**

It would ensure higher variability in choosing CED circuit design methods and adjusting the key performance indicators of self-checking discrete device implementations.

Combining the codeword sets of constant-weight *r/n* and (*n* – *r*)/*n* codes, where $r \in \left\{1,\ 2, ..., \left\lfloor \frac{n}{2} \right\rfloor\right\}$ is the weight of a codeword, yields a composition of the pair of constant-weight codes with the desired property of their codewords. The binomial coefficients corresponding to such pairs of constant-weight codes are highlighted by the same color on Pascal's triangle in Fig. 1b. The non-highlighted ones are the codes with *r* = 0, the degenerate pairs (for which the weight is $r = \frac{n}{2}$ for even *n*), and the pairs of codes with the weights differing by 1.

**Proposition 1.** *A composition of pairs of r/n and (n – r)/n codes will detect any single errors in codewords if*

$$n - 2r \geq 2. \quad (1)$$

P r o o f. The codewords of an *r/n* code have weight *r*; the codewords of an (*n* – *r*)/*n* code, weight *n* – *r*. Clearly, *r/n* and (*n* – *r*)/*n* codes will detect any single errors distorting codewords belonging to the given code into those of the same code. Only single errors distorting codewords of an *r/n* code into those of an (*n* – *r*)/*n* code may be undetected. This is possible only if the weight of the codewords of an (*n* – *r*)/*n* code exceeds that of an *r/n* code by 1. In other words, $(n-r)-r=1$, which is equivalently written as $n-2r=1$. Hence, inequality (1) is valid for detecting any single errors. ♦

Thus, we obtain a whole group of modified codes based on pairs of constant-weight codes. They can be applied to design some controllable and self-checking discrete devices as well as to provide their diagnostic support.

## 2. CHARACTERISTICS OF ERROR DETECTION BY COMPOSITIONS OF TWO CONSTANT-WEIGHT CODES

We determine the number of errors that will not be detected by compositions of two constant-weight codes as well as characterize them.

**Proposition 2.** *The total number of codeword errors undetectable by a selected composition of pairs of r/n and (n – r)/n codes is given by*

$$N_{ND} = 2C_n^r\left(2C_n^r - 1\right). \quad (2)$$

P r o o f. The codeword set of an *r/n* code has cardinality $C_n^r$; the codeword set of an $(n-r)/n$ code, cardinality $C_n^{n-r} = C_n^r$. Hence, the composition is formed by $2C_n^r$ codewords. An error will not be detected if and only if it distorts a codeword of a given composition of codes into a codeword belonging to codes from this composition. For each codeword, there are $2C_n^r - 1$ such distortions in total. Therefore, we arrive at the expression (2). ♦

Let (*r/n* + (*n* – *r*)/*n*) codes denote a composition of pairs of the corresponding constant-weight codes. For the first values of *n* as an example, they are given in Table 1. Here, the color indicates constant-weight $\frac{n}{2}/n$ codes, which also have pairs of orthogonal combinations over all bits in the codeword set.

Table 2 characterizes errors undetectable by (*r/n* + (*n* – *r*)/*n*) codes. For each code, two numbers are

*Table 1*

**Compositions of pairs of constant-weight codes**

| *n* | *r* | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 4 | 1/4 + 3/4 | 2/4 | – | – | – | – |
| 5 | 1/5 + 4/5 | – | – | – | – | – |
| 6 | 1/6 + 5/6 | 2/6 + 4/6 | 3/6 | – | – | – |
| 7 | 1/7 + 6/7 | 2/7 + 5/7 | – | – | – | – |
| 8 | 1/8 + 7/8 | 2/8 + 6/8 | 3/8 + 5/8 | 4/8 | – | – |
| 9 | 1/9 + 8/9 | 2/9 + 7/9 | 3/9 + 6/9 | – | – | – |
| 10 | 1/10 + 9/10 | 2/10 + 8/10 | 3/10 + 7/10 | 4/10 + 6/10 | 5/10 | – |
| 11 | 1/11 + 10/11 | 2/11 + 9/11 | 3/11 + 8/11 | 4/11 + 7/11 | – | – |
| 12 | 1/12 + 11/12 | 2/12 + 10/12 | 3/12 + 9/12 | 4/12 + 8/12 | 5/12 + 7/12 | 6/12 |
| 13 | 1/13 + 12/13 | 2/13 + 11/13 | 3/13 + 10/13 | 4/13 + 9/13 | 5/13 + 8/13 | – |
| 14 | 1/14 + 13/14 | 2/14 + 12/14 | 3/14 + 11/14 | 4/14 + 10/14 | 5/14 + 9/14 | 6/14 + 8/14 |
| 15 | 1/15 + 14/15 | 2/15 + 13/15 | 3/15 + 12/15 | 4/15 + 11/15 | 5/15 + 10/15 | 6/15 + 9/15 |

*Table 2*

**Characterization of errors undetectable by (*r*/*n* + (*n* – *r*)/*n*) codes**

| *n* | The number of undetectable errors and their share in the total number of errors in codewords | | | | | |
|---|---|---|---|---|---|---|
| | *r* = 1 | *r* = 2 | *r* = 3 | *r* = 4 | *r* = 5 | *r* = 6 |
| 4 | 56<br>23.33333% | 30<br>12.5% | – | – | – | – |
| 5 | 90<br>9.07258% | – | – | – | – | – |
| 6 | 132<br>3.27381% | 870<br>21.57738% | 380<br>9.4246% | – | – | – |
| 7 | 182<br>1.11959% | 1722<br>10.59301% | – | – | – | – |
| 8 | 240<br>0.36765% | 3080<br>4.71814% | 12 432<br>19.04412% | 4830<br>7.3989% | – | – |
| 9 | 306<br>0.11696% | 5112<br>1.95389% | 28 056<br>10.72346% | – | – | – |
| 10 | 380<br>0.03628% | 8010<br>0.76464% | 57 360<br>5.47562% | 175 980<br>16.79917% | 63 252<br>6.03808% | – |
| 11 | 462<br>0.01102% | 11 990<br>0.286% | 108 570<br>2.58978% | 434 940<br>10.37484% | – | – |
| 12 | 552<br>0.00329% | 17 292<br>0.10309% | 193 160<br>1.1516% | 979 110<br>5.83738% | 2 507 472<br>14.94935% | 852 852<br>5.08464% |
| 13 | 650<br>0.00097% | 24 180<br>0.03604% | 326 612<br>0.48675% | 2 043 470<br>3.04538% | 6 622 902<br>9.8701% | – |
| 14 | 756<br>0.00028% | 32 942<br>0.01227% | 529 256<br>0.19718% | 4 006 002<br>1.49244% | 16 028 012<br>5.97126% | 36 066 030<br>13.43646% |
| 15 | 870<br>0.00008% | 43 890<br>0.00409% | 827 190<br>0.07704% | 7 450 170<br>0.69387% | 36 066 030<br>3.35901% | 100 190 090<br>9.33121% |

specified (top and bottom in cells), namely, the number of undetectable errors and the share of undetectable errors in the total number of errors in the codewords ($\gamma_n$, in %). Compositions of two constant-weight codes detect a significant number of errors. With increasing *n* for each fixed *r*, the share of undetectable errors gradually drops (Fig. 2).

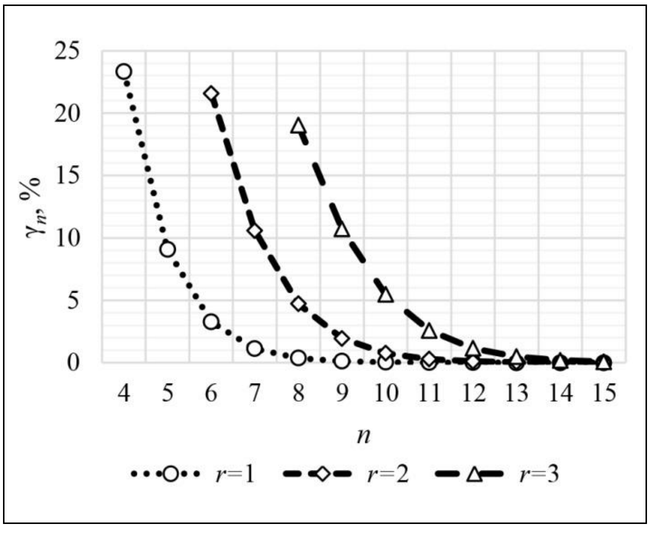


**Fig. 2. The dependences of $\gamma_n$ on *n* for different (*r*/*n* + (*n* – *r*)/*n*) codes.**

Now we characterize the errors undetectable by different (*r*/*n* + (*n* – *r*)/*n*) codes by their types (unidirectional, symmetrical and asymmetrical) and multiplicities *d*.

First, consider (1/*n* + (*n* – 1)/*n*) codes.

Let us take the first (1/4 + 3/4) code and choose an arbitrary codeword of the 1/4 code, e.g., <0001>. It can be distorted into codewords belonging to the composition of these codes in $2C_4^1 - 1 = 7$ ways. The number of distortions of the codeword <0001> into the codewords belonging to the 1/4 code is $C_4^1 - 1 = 3$. Since the codeword weight is preserved, all these distortions will be double symmetrical distortions. The codeword <0001> can be distorted into codewords belonging to the 3/4 code in $C_4^3 = 4$ ways. Moreover, if after the distortion the single 1 in the codeword preserves its value, the error will be unidirectional. As the codewords of the 3/4 code have weight *n* – *r* = 3, there are $C_{4-1}^{3-1} = C_3^2 = 3$ possible ways of such distortions. The multiplicity of unidirectional undetectable errors will equal 2. The remaining single codeword <1110> of the 3/4 code is orthogonal over all bits to the codeword <0001>. The latter can be distorted into the former only under a quadruple asymmetrical error.

The same fact takes place for the other codewords of (1/4 + 3/4) codes.

Proceeding to the second (1/5 + 4/5) code, we choose an arbitrary codeword of the 1/5 code, e.g., <00001>. It can be distorted into codewords belonging to the composition of these codes in $2C_5^1 - 1 = 9$ ways. The number of distortions of the codeword <00001> into the codewords belonging to the 1/5 code is $C_5^1 - 1 = 4$. Due to preserving the codeword weight, all these distortions will be double symmetrical distortions. The codeword <00001> can be distorted into codewords belonging to the 4/5 code in $C_5^4 = 5$ ways. Moreover, if after the distortion the single 1 in the codeword preserves its value, the error will be unidirectional. As the codewords of the 4/5 code have weight $n - r = 4$, there are $C_{5-1}^{4-1} = C_4^3 = 4$ possible ways of such distortions. The multiplicity of unidirectional undetectable errors will equal 3. The remaining single codeword <11110> of the 4/5 code is orthogonal over all bits to the codeword <00001>. The latter can be distorted into the former only under a quintuple asymmetrical error. The other codewords of the (1/5 + 4/5) codes are considered by analogy.

Continuing the series of considerations, we arrive at the following results, valid for $(1/n + (n-1)/n)$ codes:

– For each codeword, there are $n - 1$ possible ways to distort it into a codeword belonging only to this code, characterized by a double symmetrical error.

– For each codeword, there are $n - 1$ possible ways to distort it into a codeword belonging to the second code in the composition, characterized by an $(n - 2r) = (n - 2)$-tuple unidirectional error.

– For each codeword, there is a single possible way to distort it into a codeword belonging to the second code in the composition, characterized by an $n$-tuple asymmetrical error.

There exist $2C_n^1 = 2n$ codewords in total; therefore, each of the above numbers, for each error type and multiplicity, should be multiplied by this value to obtain the total number of undetectable errors by type and multiplicity.

Let us represent the total number of undetectable errors as

$$N_{ND} = N_{\upsilon,d}\upsilon_d + N_{\sigma,d}\sigma_d + N_{\alpha,d}\alpha_d, \qquad (3)$$

with the following notation: $N_{\upsilon,d}$, $N_{\sigma,d}$, and $N_{\alpha,d}$ are the numbers of undetectable unidirectional, symmetrical, and asymmetrical errors, respectively, with multiplicity $d$; the symbols $\upsilon_d$, $\sigma_d$, and $\alpha_d$ indicate the belonging of undetectable errors to the class of unidirectional, symmetrical, and asymmetrical errors, respectively, with multiplicity $d$.

Using the expression (3) and the above reasoning, we derive the following formula for undetectable errors for $(1/n + (n-1)/n)$ codes:

$$\begin{aligned} N_{ND}^{1/n+(n-1)/n} &= 2n\big((n-1)\upsilon_{n-2} + (n-1)\sigma_2 + 1\alpha_n\big) \\ &= 2n(n-1)(\upsilon_{n-2} + \sigma_2) + 2n\alpha_n. \end{aligned} \qquad (4)$$

For example, for (1/4 + 3/4) codes, formula (4) gives

$$\begin{aligned} N_{ND}^{1/4+3/4} &= 2\cdot 4\cdot(4-1)(\upsilon_2 + \sigma_2) + 2\cdot 4\alpha_4 \\ &= 24\upsilon_2 + 24\sigma_2 + 8\alpha_4. \end{aligned}$$

Now we generalize this result to the composition of $(2/n + (n-2)/n)$ codes.

As an example, let us take the (2/6 + 4/6) code and choose an arbitrary codeword of the 2/6 code, e.g., <000011>. There are $2C_6^2 - 1 = 29$ its possible distortions into codewords belonging to this composition. Among them, there are $C_6^2 - 1 = 14$ distortions into the codewords of the 2/6 code due to double symmetrical errors. Also, there are $C_6^2 = 15$ distortions into the codewords of the 4/6 code. Such errors have the following structure. There is a single distortion into the codeword of the 4/6 code orthogonal over all bits to the codeword <111100>, due to a sextuple asymmetrical error. There are $C_{6-2}^2 = C_4^2 = 6$ distortions under which all 1s in the codeword <000011> preserve their positions and two 0s are distorted. Such distortions are caused by double unidirectional errors. Also, there are distortions under which a single 1 preserves its value in the codeword <000011>, a single 1 is distorted, and three 0s are distorted. The number of possible distortions of 1s and 0s described above are $C_2^1$ and $C_4^3$, respectively. In total, we have $C_4^3 C_2^1 = 8$ quadruple asymmetrical distortions.

This error structure is inherent in all codewords in the composition under consideration.

It is easy to generalize the results to $(2/n + (n-2)/n)$ codes:

– For each codeword, there are $C_n^2 - 1$ possible ways to distort it into a codeword belonging only to this code, characterized by a double symmetrical error.

– For each codeword, there are $C_{n-2}^2$ possible ways to distort it into a codeword belonging to the second code in the composition, characterized by an $(n - 2r) = (n - 4)$-tuple unidirectional error.

– For each codeword, there is a single possible way to distort it into a codeword belonging to the second code in the composition, characterized by an $n$-tuple asymmetrical error.

There exist $2C_n^2$ codewords in total; therefore, each of the above numbers, for each error type and multiplicity, should be multiplied by this value to obtain the total number of undetectable errors by type and multiplicity:

$$N_{ND}^{2/n+(n-2)/n} = 2C_n^2\left(C_{n-2}^2\upsilon_2 + \left(C_n^2 - 1\right)\sigma_2 + 2C_{n-2}^{n-3}\alpha_4 + 1\alpha_n\right) = 2C_n^2 C_{n-2}^2\upsilon_2 + 2C_n^2\left(C_n^2 - 1\right)\sigma_2 + 4C_n^2 C_{n-2}^{n-3}\alpha_4 + 2C_n^2\alpha_n. \quad (5)$$

For example, for (2/6 + 4/6) codes, formula (5) gives

$$N_{ND}^{2/6+4/6} = 2C_6^2 C_4^2\upsilon_2 + 2C_6^2\left(C_6^2 - 1\right)\sigma_2 + 4C_6^2 C_4^3\alpha_4 + 2C_6^2\alpha_6 = 180\upsilon_2 + 420\sigma_2 + 240\alpha_4 + 30\alpha_6.$$

The generalization to $(r/n + (n - r)/n)$ codes leads to the following:

– For each codeword, there are $C_n^r - 1$ possible ways to distort it into a codeword belonging only to this code, characterized by a double symmetrical error.

– For each codeword, there are $C_{n-r}^r$ possible ways to distort it into a codeword belonging to the second code in the composition, characterized by an $(n - 2r)$-tuple unidirectional error.

– For each codeword, there are

$$C_r^1 C_{n-r}^{(n-r)-(r-1)} + C_r^2 C_{n-r}^{(n-r)-(r-2)} + \ldots + C_r^{r-1} C_{n-r}^{n-r-(r-(r-1))} + C_r^r C_{n-r}^{n-r-(r-r)} = C_r^1 C_{n-r}^{n-2r+1} + C_r^2 C_{n-r}^{n-2r+2} + \ldots + C_r^{r-1} C_{n-r}^{n-r-1} + C_r^r C_{n-r}^{n-r} = \sum_{i=1}^{r} C_r^i C_{n-r}^{(n-r)-(r-i)}$$

possible ways to distort it into a codeword belonging to the second code in the composition, characterized by asymmetrical errors. Note that the first term is the number of asymmetrical errors of multiplicity $n - 2r + 2$; the second term, the number of asymmetrical errors of multiplicity $n - 2r + 4$; ....; the $(r - 1)$th term, the number of asymmetrical errors of multiplicity $((n - r) - (r - (r - 1)) + (r - 1)) = (n - 2)$; finally, the $r$th term, the number of asymmetrical errors of multiplicity $n$.

The total number of codewords in the composition considered is $2C_n^r$.

Thus, we arrive at the general formula for the number of errors of different types and multiplicities undetectable by $(r/n + (n - r)/n)$ codes:

$$N_{ND}^{r/n+(n-r)/n} = 2C_n^r\left(C_{n-r}^r\upsilon_{n-2r} + \left(C_n^r - 1\right)\sigma_2 + C_r^1 C_{n-r}^{n-2r+1}\alpha_{n-2r+2} + C_r^2 C_{n-r}^{n-2r+2}\alpha_{n-2r+4} + \ldots + C_r^{r-1} C_{n-r}^{n-r-1}\alpha_{n-2} + C_r^r C_{n-r}^{n-r}\alpha_n\right)$$
$$= 2C_n^r C_{n-r}^r\upsilon_{n-2r} + 2C_n^r\left(C_n^r - 1\right)\sigma_2 + 2C_n^r C_r^1 C_{n-r}^{n-2r+1}\alpha_{n-2r+2} + 2C_n^r C_r^2 C_{n-r}^{n-2r+2}\alpha_{n-2r+4} + \ldots + 2C_n^r C_r^{r-1} C_{n-r}^{n-r-1}\alpha_{n-2} + 2C_n^r C_r^r C_{n-r}^{n-r}\alpha_n. \quad (6)$$

Formulas (4) and (5) are special cases of (6).

The error detection characteristics of compositions of constant-weight codes can be used in practice to build discrete devices with fault detection.

## 3. COMPOSITIONS OF AN EVEN NUMBER OF CONSTANT-WEIGHT CODES

For $n \geq 8$, compositions of more than two constant-weight codes can be constructed in which the codeword set will contain only pairs of codewords orthogonal over all bits. Such codes are constructed for even values of $n \geq 8$. Desired combinations can be formed by uniting the codeword sets of the four constant-weight codes highlighted on Pascal's triangle (Fig. 3).

Table 3 shows the composition of quadruples of constant-weight codes, and Table 4 characterizes the errors undetectable by them. For $n > 8$, compositions of quadruple constant-weight codes do not detect hundredths and thousandths of a percent of the total number of errors occurring in codewords with the corresponding codeword length.

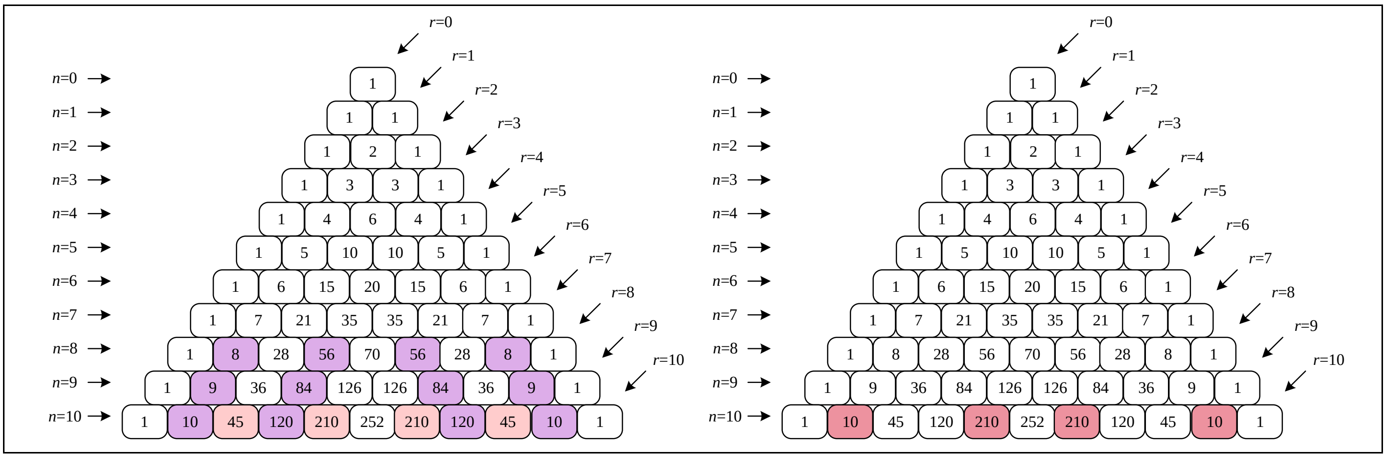


**Fig. 3. Ways to select four constant-weight codes with orthogonal combinations over all bits on Pascal's triangle.**

*Table 3*

**Compositions of quadruples of constant-weight codes**

| $n$ | $r_1, r_2$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $r_1$=1, $r_2$=3 | $r_1$=1, $r_2$=4 | $r_1$=1, $r_2$=5 | $r_1$=1, $r_2$=6 | $r_1$=2, $r_2$=4 | $r_1$=2, $r_2$=5 | $r_1$=2, $r_2$=6 | $r_1$=3, $r_2$=5 | $r_1$=3, $r_2$=6 | $r_1$=4, $r_2$=6 |
| 8 | 1/8+7/8, 3/8+5/8 | – | – | – | – | – | – | – | – | – |
| 9 | 1/9+8/9, 3/9+6/9 | – | – | – | – | – | – | – | – | – |
| 10 | 1/10+9/10, 3/10+7/10 | 1/10+9/10, 4/10+6/10 | – | – | 2/10+8/10, 4/10+6/10 | – | – | – | – | – |
| 11 | 1/11+10/11, 3/11+8/11 | 1/11+10/11, 4/11+7/11 | – | – | 2/11+9/11, 4/11+7/11 | – | – | – | – | – |
| 12 | 1/12+11/12, 3/12+9/12 | 1/12+11/12, 4/12+8/12 | 1/12+11/12, 5/12+7/12 | – | 2/12+10/12, 4/12+8/12 | 2/12+10/12, 5/12+7/12 | – | 3/12+9/12, 5/12+7/12 | – | – |
| 13 | 1/13+12/13, 3/13+10/13 | 1/13+12/13, 4/13+9/13 | 1/13+12/13, 5/13+8/13 | – | 2/13+11/13, 4/13+9/13 | 2/13+11/13, 5/13+8/13 | – | 3/13+10/13, 5/13+8/13 | – | – |
| 14 | 1/14+13/14, 3/14+11/14 | 1/14+13/14, 4/14+10/14 | 1/14+13/14, 5/14+9/14 | 1/14+13/14, 6/14+8/14 | 2/14+12/14, 4/14+10/14 | 2/14+12/14, 5/14+9/14 | 2/14+12/14, 6/14+8/14 | 3/14+11/14, 6/14+8/14 | 3/14+11/14, 6/14+8/14 | 4/14+10/14, 6/14+8/14 |
| 15 | 1/15+14/15, 3/15+12/15 | 1/15+14/15, 4/15+11/15 | 1/15+14/15, 5/15+10/15 | 1/15+14/15, 6/15+9/15 | 2/15+13/15, 4/15+11/15 | 2/15+13/15, 5/15+10/15 | 2/15+13/15, 6/15+9/15 | 3/15+12/15, 6/15+9/15 | 3/15+12/15, 6/15+9/15 | 4/15+11/15, 6/15+9/15 |

*Table 4*

**Characteristics of errors undetectable by compositions of quadruples of constant-weight codes**

| $n$ | The number of undetectable errors and their share in the total number of errors in codewords | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $r_1$=1, $r_2$=3 | $r_1$=1, $r_2$=4 | $r_1$=1, $r_2$=5 | $r_1$=1, $r_2$=6 | $r_1$=2, $r_2$=4 | $r_1$=2, $r_2$=5 | $r_1$=2, $r_2$=6 | $r_1$=3, $r_2$=5 | $r_1$=3, $r_2$=6 | $r_1$=4, $r_2$=6 |
| 8 | 128<br>0.19608 % | – | – | – | – | – | – | – | – | – |
| 9 | 186<br>0.07109 % | – | – | – | – | – | – | – | – | – |
| 10 | 260<br>0.02482 % | 440<br>0.042 % | – | – | 510<br>0.04868 % | – | – | – | – | – |
| 11 | 352<br>0.0084 % | 682<br>0.01627 % | – | – | 770<br>0.01837 % | – | – | – | – | – |
| 12 | 464<br>0.00277 % | 1014<br>0.00605 % | 1608<br>0.00959 % | – | 1122<br>0.00669 % | 1716<br>0.01023 % | – | 2024<br>0.01207 % | – | – |
| 13 | 598<br>0.00089 % | 1456<br>0.00217 % | 2600<br>0.00387 % | – | 1586<br>0.00236 % | 2730<br>0.00407 % | – | 3146<br>0.00469 % | – | – |
| 14 | 756<br>0.00028 % | 2030<br>0.00076 % | 4032<br>0.0015 % | 6034<br>0.00225 % | 2184<br>0.00081 % | 4186<br>0.00156 % | 6188<br>0.00231 % | 4732<br>0.00176 % | 6734<br>0.00251 % | 8008<br>0.00298 % |
| 15 | 940<br>0.00009 % | 2760<br>0.00026 % | 6036<br>0.00056 % | 10040<br>0.00094 % | 2940<br>0.00027 % | 6216<br>0.00058 % | 10220<br>0.00095 % | 6916<br>0.00064 % | 10920<br>0.00102 % | 12740<br>0.00119 % |

The above considerations for determining the error detection characteristics of pairs of compositions of constant-weight codes by error types and multiplicities (see the previous section) can be repeated to characterize the errors undetectable by the compositions of quadruple constant-weight codes.

## 4. APPLICATION OF COMPOSITIONS OF TWO CONSTANT-WEIGHT CODES IN SELF-CHECKING DISCRETE DEVICE DESIGN

Figure 4 shows the structural diagram of a self-checking discrete device with a CED circuit based on an $(r/n + (n - r)/n)$ code. It uses two diagnostic attributes for computation control: the belonging of the codewords formed to a given $(r/n + (n - r)/n)$ code and the belonging of each function describing the check bits to the class of self-dual functions.

The object under diagnosis is a combinational discrete device $F(X)$ that calculates the values of Boolean functions $f_1(X)$, $f_2(X)$,…, $f_{n-1}(X)$, $f_n(X)$ when sets of argument values $\langle x_t\ x_{t-1}\ ...\ x_2\ x_1\rangle = \langle X\rangle$ are supplied to its inputs. The device $F(X)$ is augmented with a special CED circuit to control computation. In addition, the operation of this circuit is adjusted appropriately [27].

The outputs of the device $F(X)$ are connected to the inputs of a Boolean signal correction (BSC) block in the CED circuit. This block is intended to transform the output signals of the device $F(X)$ into signals $h_1(X)$, $h_2(X)$,…, $h_{n-1}(X)$, $h_n(X)$ on each set of argument values. The signal correction block is formed by a cascade of two-input XORs: their first inputs receive the signals $f_1(X)$, $f_2(X)$,…, $f_{n-1}(X)$, $f_n(X)$ whereas their second inputs the signals $g_1(X)$, $g_2(X)$,…, $g_{n-1}(X)$, $g_n(X)$ with the same subscripts from a block $G(X)$ calculating the Boolean correction functions:

$$h_i\left(X\right)=f_i\left(X\right)\oplus g_i\left(X\right), i=\overline{1,n}. \qquad (7)$$

The transformation (7) is performed on each set of argument values to endow the functions $h_1(X)$, $h_2(X)$,…, $h_{n-1}(X)$, $h_n(X)$ with the following two properties. First, the codeword $\langle h_n(X)\ h_{n-1}(X)\ …\ h_2(X)\ h_1(X)\rangle$ must belong to the $(r/n + (n - r)/n)$ code selected. Second, the values of each function $h_1(X)$, $h_2(X)$,…, $h_{n-1}(X)$, $h_n(X)$ on each set of argument values are generated so that the functions turn out to be self-dual. This requires only that orthogonal combinations over all bits belonging to the $(r/n + (n - r)/n)$ code be formed on orthogonal sets of argument values (see above). It is necessary to redefine the values of the vectors $\langle h_n(X)\ h_{n-1}(X)\ …\ h_2(X)\ h_1(X)\rangle$ on all sets of argument values so that all codewords belonging to the $r/n$ code and $(n - r)/n$ code will be formed at least once. This feature requires the constraint

$$t \ge \left\lceil \log_2 2C_n^r \right\rceil.$$

The belonging of the codewords $\langle h_n(X)\ h_{n-1}(X)\ …\ h_2(X)\ h_1(X)\rangle$ to the $(r/n + (n - r)/n)$ code selected is verified by a totally self-checking checker (TSC). Such checkers can be designed in various ways, e.g., by separating two functions $z_1^0\left(X\right)$ and $z_1^1\left(X\right)$, each being the disjunction of conjunctions corresponding to the codewords of the $r/n$ code and $(n - r)/n$ code, respectively. (By the way, they may appear in each of the functions, but only once!) It is possible to use bracketed forms to simplify

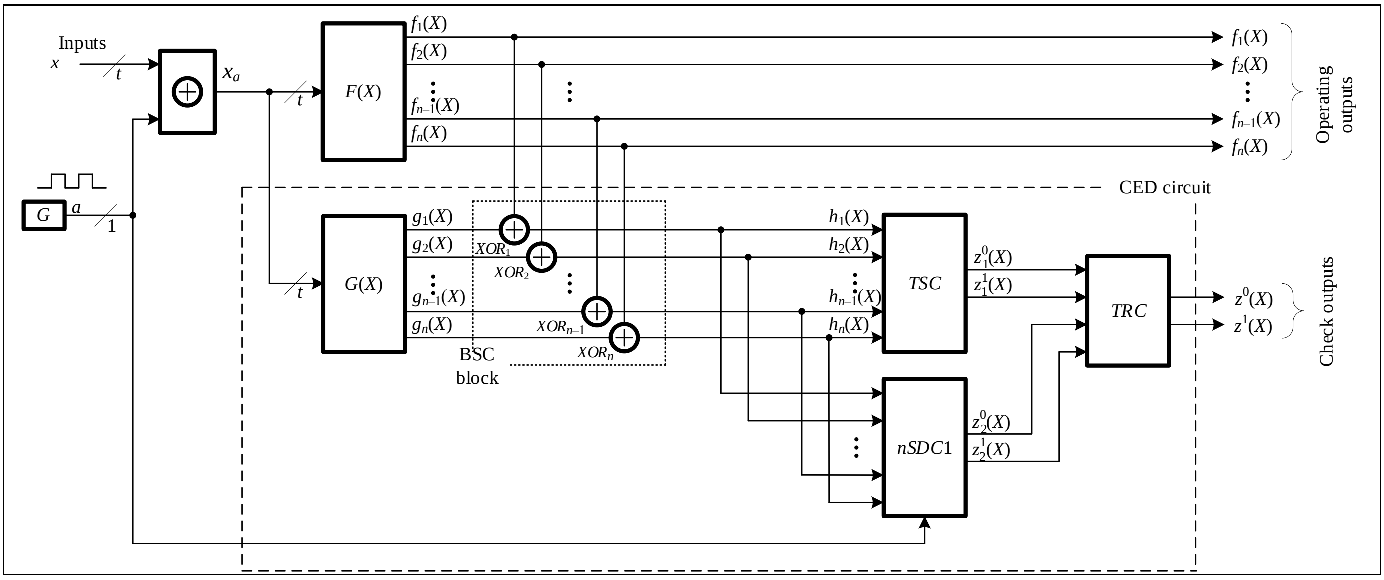


**Fig. 4. The structural diagram of a self-checking discrete device.**

checker structures. The design methods of constant-weight code checkers described in [9, 10] are not directly applicable to build checkers of their compositions.

The belonging of each function $h_1(X)$, $h_2(X)$,…, $h_{n-1}(X)$, $h_n(X)$ to the class of self-dual Boolean functions is checked using *nSDC*1, a device that receives *n* self-dual signals at its inputs and produces the values of two functions $z_2^0(X)$ and $z_2^1(X)$ at its outputs. Such a device can be implemented in two ways. The first solution is to install *n* elementary self-dual checkers (SDC) [27] and $n - 1$ elementary two-rail checkers (TRC) [28]. The second is based on the preliminary compression of *n* self-dual signals into one signal using three-input XORs (see the corresponding scheme in [29]) and control of the received signal using one SDC device.

The outputs $z_1^0(X)$ and $z_1^1(X)$, as well as $z_2^0(X)$ and $z_2^1(X)$, operate in the two-rail logic. Violation of the two-rail principle indicates a computational error, indirectly meaning the presence of faults in one of the blocks. The outputs $z_1^0(X)$ and $z_1^1(X)$, as well as $z_2^0(X)$ and $z_2^1(X)$, are connected to the inputs of one TRC block to produce a single reference signal $Z^0(X)$ and $Z^1(X)$.

According to the control principles for self-dual signals, it is necessary to compare sequentially the value on the direct combination supplied and on the orthogonal one over all bits, which actually reduces the performance twofold [30]. Then the first combination supplied is the operating one, and the second is the check combination. Thus, it is required to supply, to the inputs of the device, pairs of sets of argument values composed of operating combinations and check combinations orthogonal over all variables to the former. This is conveniently organized in the pulse operation mode of the structure shown in Fig. 4. The mode is implemented by installing a generator *G* of rectangular pulses with frequency $S = 2$, which produces the sequence *a* 0101...01 at its output. This sequence is supplied to the second inputs of two-input XORs, installed for each input $x_1, x_2, ..., x_{t-1}, x_t$. The first inputs of XORs are fed with the values of the input variables themselves. With such a circuit, the Boolean zero signal is transformed into the sequence 0101...01 and the Boolean one signal into the sequence 1010...10. Thus, the sets of argument values arrive at the inputs of the block $F(X)$ in pairs, namely, the operating and check ones, the latter being orthogonal over all bits to the former. Also, the signal from the rectangular pulse generator is necessary for the operation of the checkers of self-dual signals (SDC) in the CED circuit.

Separate examples of CED circuit design by the above method are beyond the scope of this paper. Note that in practice, the use of $(r/n + (n - r)/n)$ codes in the CED circuit design according to Fig. 4 yields totally self-checking discrete devices with different performance indicators. First of all, it is important to cover errors at the outputs of the device $F(X)$, ensure the self-checking property of the CED circuit, and obtain the simplest CED circuit implementation in order to design less redundant structures compared to the duplication method. Given a high variability in obtaining the values of the functions $g_1(X)$, $g_2(X)$,…, $g_{n-1}(X)$, $g_n(X)$ on each set of argument values, it is possible to form sufficiently many structures and choose among them the simplest in terms of complexity. This indicator determines the device complexity [31]. For instance, the number of ways to redefine the function values essentially depends on the number *t* of inputs in the object under diagnosis $F(X)$. It suffices to redefine the function values on the first half of the sets, their number equals $2^{t-1}$. On each of these sets, redefinition can be performed in $2C_n^r$ ways. On the second half of the sets of argument values, the function values are obtained based on the properties of self-dual functions: it is required to redefine the functions to opposite values on the sets orthogonal over all bits. In total, there are $2^t C_n^r$ ways to redefine the function values realized at the outputs of the block $G(X)$. Given many ways to build the blocks $G(X)$, one can choose the simplest implementations while ensuring the testability of CED circuit components. In many practical cases, it is possible to implement the block $G(X)$ simpler than $F(X)$, which significantly affects the hardware redundancy of CED circuits and allows building less redundant self-checking devices compared to duplication with improved controllability indicators [23]. Nevertheless, for some blocks $F(X)$, it is impossible to build a simpler implementation of this structure than in the case of duplication, which is a disadvantage of CED circuits with correction of all signals from the object under diagnosis [32]. According to studies, under certain conditions, the number of outputs of the block $G(X)$ can be reduced by using the same functions for the correction of different signals from the block $F(X)$. This approach works only in particular cases and has been underinvestigated so far. However, it can be considered promising for improving CED circuit design algorithms based on the structure in Fig. 4.

## CONCLUSIONS

Using compositions of two constant-weight codes, *r*/*n* and $(n - r)/n$ with $r \in \left\{1, 2, ..., \left\lfloor \frac{n}{2} \right\rfloor\right\}$ and condition (1) allows significantly expanding the number of ways to organize controllable and self-checking discrete devices with computation control via two diagnostic attributes in comparison with the application of only $\frac{n}{2}/n$ codes with even $n$. Compositions of constant-weight codes with orthogonal combinations over all bits can be separated out for $n \geq 4$. With increasing $n$, the number of codewords in such codes grows accordingly, and the number of ways to select exactly two constant-weight codes sequentially increases by 1 for each even $n$, starting from $n = 4$. For $n \geq 8$, it is possible to construct a composition of an even number of constant-weight codes with orthogonal combinations over all bits.

In a wide class of compositions of constant-weight codes with orthogonal combinations over all bits, an appropriate composition can be selected for a given number of outputs of the object under diagnosis and for a possible separation of groups of outputs with their smaller number to organize separate computation control subcircuits. The application of compositions of constant-weight codes with small $n$ may be of interest due to simpler checkers and simpler conditions for ensuring their total self-checking property.

Further research can be connected with developing CED circuit design methods based on compositions of constant-weight codes with minimization of various quality indicators (structural redundancy, controllability, power consumption, etc.). Practical applications of these codes can be interesting, especially in the areas of science and technology with rarely changing input data. (Such systems are widespread and include, e.g., air defense systems, electrical interlocking systems in railway transport, control systems in the nuclear industry, etc.). In these systems, operational diagnosis and the readiness of devices are very topical tasks [33, 34].

The principles to extract compositions of constant-weight codes described in this paper, as well as their properties, may be in demand when building highly reliable discrete systems on modern and promising components [35, 36].

## REFERENCES


1. 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
2. Tallini, L.G. and Bose, B., Design of Balanced and Constant Weight Codes for VLSI systems, *IEEE Transactions on Computers*, 1998, vol. 47, no. 5, pp. 556–572. DOI: 10.1109/12.677239
3. Butler, J.T. and Sasao, T., Index to Constant Weight Codeword Converter, *Proceedings of 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Seoul, South Korea, 2011. DOI: 10.1109/MWSCAS.2011.6026312
4. Samofalov, K.G., Romankevich, A.M., Valuiskii, V.N., et al., *Prikladnaya teoriya tsifrovykh avtomatov* (Applied Theory of Digital Automata), Samofalov, K.G., Ed., Kiev: Vishcha Shkola, 1987. (In Russian.)
5. 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.)
6. Göessel, M., Ocheretny, V., Sogomonyan, E., and Marienfeld, D., *New Methods of Concurrent Checking*, 1st ed., Dordrecht: Springer Science+Business Media, 2008.
7. Das, D.K., Roy, S.S., Dmitiriev, A., et al. Constraint Don't Cares for Optimizing Designs for Concurrent Checking by 1-out-of-3 Codes, *Proceedings of the 10th International Workshops on Boolean Problems*, Freiberg, Germany, 2012, pp. 33–40.
8. Sapozhnikov, V.V. and Sapozhnikov, Vl.V., Self-Checking Checkers for Constant-weight codes, *Autom. Remote Control*, 1992, vol. 53, no. 3, pp. 321–348.
9. Sapozhnikov, V.V. and Sapozhnikov, Vl.V., *Samoproveryaemye diskretnye ustroistva* (Self-Checking Discrete Devices), St. Petersburg: Energoatomizdat, 1992. (In Russian.)
10. Piestrak, S.J., *Design of Self-Testing Checkers for Unidirectional Error Detecting Codes*, Wrocław: Oficyna Wydawnicza Politechniki Wrocłavskiej, 1995.
11. Tarnik, M., Design of Embedded Constant Weight Code Checkers Based on Averaging Operations, *Proceedings of 2010 IEEE 16th International On-Line Testing Symposium*, Corfu, Greece, 2010. DOI: 10.1109/IOLTS.2010.5560193
12. Matrosova, A.Yu., Butorina, N.B., and Yakimova, N.O., Checker Design Based on Monotonous Function Implementation, *Izvestiya Vuzov. Fizika*, 2013, vol. 56, no. 9-2, pp. 171–173. (In Russian.)
13. Efanov, D.V., Some Features of Error Detection by Uniform Indivisible Codes, *Journal of Instrument Engineering*, 2019, vol. 62, no. 7, pp. 621–631. DOI: 10.17586/0021-3454-2019-62-7-621-631. (In Russian.)
14. Korshunov, A.D., Some Unsolved Problems in Discrete Mathematics and Mathematical Cybernetics, *Russian Math. Surveys*, 2009, vol. 64, no. 5, pp. 787–803. https://doi.org/10.1070/RM2009v064n05ABEH004640.
15. Mozharov, G.P., Fault-Tolerant Computer Networks Constructed on the Basis of Combinatory Block Designs, *Herald of the Bauman Moscow State Tech. Univ. Ser. Instrum. Eng.*, 2016, no. 6, pp. 41–53. DOI: 10.18698/0236-3933-2016-6-41-53. (In Russian.)
16. Chioktour, V. and Kakarountas, A., Adaptive BIST for Concurrent On-Line Testing on Combinational Circuits, *Electronics*, 2022, vol. 19, art. no. 3193. DOI: 10.3390/electronics11193193
17. Sahana, A.R., Chiraag, V., Suresh, G., et al., Application of Error Detection and Correction Techniques to Self-Checking VLSI Systems: An Overview, *Proceedings of 2023 IEEE Guwahati Subsection Conference (GCON)*, Guwahati, 2023. DOI: 10.1109/GCON58516.2023.10183449
18. Slabakov, E.V. and Sogomonyan, E.S., Design of Completely Self-Checking Combinational Circuits with the Use of Constant-weight codes, *Autom. Remote Control*, 1980, vol. 41, no. 9, pp. 1326–1333.
19. 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

20. Efanov, D.V., Sapozhnikov, V.V., and Sapozhnikov, Vl.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.
21. Efanov, D.V., Sapozhnikov, V.V., Sapozhnikov, V.V., Synthesis of Self-Checking Combination Devices Based on Allocating Special Groups of Outputs, *Autom. Remote Control*, 2018, vol. 79, no. 9, pp. 1609–1620. https://doi.org/10.1134/S0005117918090060.
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. DOI: 10.3103/S0146411620040045.
23. Efanov, D.V. and Pogodina, T.S., Properties Investigation of Self-Dual Combinational Devices with Calculation Control Based on Hamming Codes, *Informatics and Automation*, 2023, vol. 22, no. 2, pp. 349–392. DOI: 10.15622/ia.22.2.5. (In Russian.)
24. Efanov, D., Sapozhnikov, V., Sapozhnikov, Vl., et al., Self-Dual Complement Method up to Constant-Weight Codes for Arrangement of Combinational Logical Circuits Concurrent Error-Detection Systems, *Proceedings of 17th IEEE East-West Design & Test Symposium (EWDTS'2019)*, Batumi, Georgia, 2019, pp. 136–143. DOI: 10.1109/EWDTS.2019.8884398.
25. Lala, P.K., *Self-Checking and Fault-Tolerant Digital Design*, San Francisco: Morgan Kaufmann Publishers, 2001.
26. 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, *Control Sciences*, 2017, no. 1, pp. 57–64. (In Russian.)
27. Efanov, D.V. and Pivovarov, D.V., Checkers of Self-Dual and "`Close in Meaning"' Signals, *Journal of Instrument Engineering*, 2024, vol. 67, no. 1, pp. 5–19. https://doi.org/10.17586/0021-3454-2024-67-1-5-19 (In Russian.)
28. Nikolos, D., Self-Testing Embedded Two-Rail Checkers, *Journal of Electronic Testing: Theory and Applications*, 1998, vol. 12, no. 1-2, pp. 69–79. DOI: 10.1023/A:1008281822966
29. 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
30. Gessel', M., Moshanin, V.I., Sapozhnikov, V.V., and Sapozhnikov, Vl.V., Fault Detection in Self-Test Combination Circuits Using the Properties of Self-Dual Functions, *Avtomat. Telemekh.*, 1997, no. 12, pp. 193–200. (In Russian.)
31. 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.
32. 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.
33. Drozd, A., Kharchenko, V., Antoshchuk, S., et al. Checkability of the Digital Components in Safety-Critical Systems: Problems and Solutions, *Proceedings of 9th IEEE East-West Design & Test Symposium (EWDTS'2011)*, Sevastopol, Ukraine, 2011, pp. 411–416. DOI: 10.1109/EWDTS.2011.6116606.
34. Sapozhnikov, Vl.V., *Sintez sistem upravleniya dvizheniem poezdov na zheleznodorozhnykh stantsiyakh s isklyucheniem opasnykh otkazov* (Design of Train Traffic Control Systems at Railway Stations with the Elimination of Dangerous Failures), Moscow: Nauka, 2021. (In Russian.)
35. Hahanov, V., Litvinova, E., Davitadze, Z., et al., Truth Table Based Intelligent Computing, *Proceedings of 31st International Conference "Mixed Design of Integrated Circuits and Systems,"* Gdansk, Poland, 2024. DOI: 10.23919/MIXDES62605.2024.10614035
36. Meliqyan, V., Davtyan, V., Harutyuyan, A., et al., Novel Adaptive Crossover Mechanism for Genetic Algorithm on Integrated Circuit Cell Placement, *Proceedings of IEEE East-West Design & Test Symposium (EWDTS)*, Yerevan, Armenia, 2024. DOI: 10.1109/EWDTS63723.2024.10873659



*This paper was recommended for publication by V. G. Lebedev, a member of the Editorial Board.*

*Received March 10, 2025, and revised May 12, 2025. Accepted June 18, 2025.*


**Author information**


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


**Cite this paper**


Efanov, D.V., Compositions of Two Constant-Weight Codes with Orthogonal Combinations over All Bits for Self-Checking Discrete Device Design. *Control Sciences* 3, 41–51 (2025).

Original Russian Text © Efanov, D.V., 2025, published in *Problemy Upravleniya*, 2025, no. 3, pp. 49–62.





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