International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34433-34442

Functionally Complete Tolerant Elements

Sergey FeofentovichTyurin
Perm National Research Polytechnic University,
Russia, 614990, Perm, Komsomolskyprospekt, 29
Artem Vladimirovich Grekov
Perm Military Institute of Internal Troops of the Ministry of Internal Affairs of the Russian Federation,
Russia, 614112, Perm, Gremyachy Log Street, 1

Abstract- Authors put forward concept of functionally complete tolerant element. Tolerance is provided by using redundant bases, those retain functional completeness in case faults or failure occurs. Functional complete tolerant element allows to perform FPGA self-repair after faults and failures.

Keywords-basis function, faults, fault tolerance, majority element, redundancy, reliability.

1. INTRODUCTION

Apparently, theory of fault-tolerant systems started with the work of J. Von Neumann (J. Von Neumann, 1956), who had formulated the paradigm of synthesis of "reliable organisms from the unreliable components". Later, A. Avižienis suggested the fault tolerance concepts (Avižienis, 1978), and then (in affiliation with J. C. Laprie, B. Randell and C. Landwehr) he built the dependable computing methodology (Avizienis andLaprie, 1986; Avizienis*et al*., 2004).

Most of the known methods of providing system reliability are based on different kinds of redundancy, and mostly this redundancy is applied on channel (system) level; i.e. the entire system redundancy or redundancy of major subsystems is implemented (Brown, 1980; Brown, 1981; Finkelstein, 2008).

For improvement of digital equipment reliability the triple redundancy is frequently used (Shanthikumar, 1988; Sambhu *et al*., 2011; Yu, 2011). This is so called passive fault-tolerance and it provides more than 300% redundancy. For establishing a passive fault-tolerance we need three equipment channels, one majority element (ME) and three power supplies, see Figure 1.

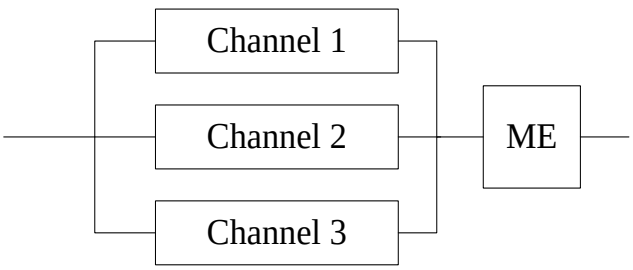

Fig 1. Using of majority element

For exponential fault model we have:

$$P = [3e^{-2\lambda t} - 2e^{-3\lambda t}], \tag{1}$$

where$P$ – probability of faultless operation, λ –failure rate of one channel, t – operation time.
With regard to majority element faults we have:

$$P = [3e^{-2\lambda t} - 2e^{-3\lambda t}] \cdot e^{-\lambda_{ME} t}, \tag{2}$$

whereλ_{ME}– failure rate of the majority element.

Generally, three majority elements are used for providing fault-tolerance, and each element passes signal to the next circuit unit, see Figure 2.

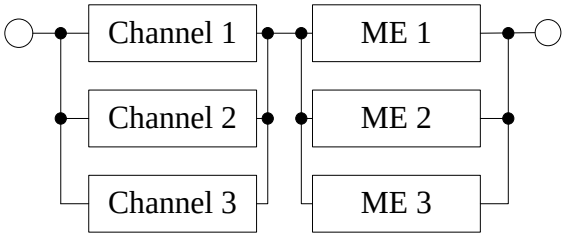

Fig 2. Using three majority elements

For three majority elements we have:

$$P = [3e^{-2\lambda t} - 2e^{-3\lambda t}][3e^{-2\lambda_{ME} t} - 2e^{-3\lambda_{ME} t}]. \tag{3}$$

When using three majority elements, the majority function is applied:

$$z_1 = k_1 k_2 \vee k_1 k_3 \vee k_2 k_3,$$
$$k_1 = k_2 = k_3 = k, \tag{4}$$

where$k$ – information from three computation channels.
This function is tolerant to single fault (failure) in one of three channels:

$$k_i \to (\overline{k_i}), k_i \to 1, k_i \to 0. \tag{5}$$

Actually, function provides truth-preserving and false-preserving.
For example, tolerance to truth constant in channel №1 is provided as follows:

$$z_1 = 1k_2 \vee 1k_3 \vee k_2 k_3 = k \vee k \vee kk = k,$$
$$k_1 = k_2 = k_3 = k. \tag{6}$$

Tolerance to false constant in channel №1 is provided as follows:

$$z_1 = 0k_2 \vee 0k_3 \vee k_2k_3 = 0 \vee 0 \vee kk = k,$$
$$k_1 = k_2 = k_3 = k. \quad (7)$$

Besides, tolerance to inversion (failure) in channel №1 is also provided:

$$z_1 = \bar{k}k \vee \bar{k}k \vee kk = 0 \vee 0 \vee kk = k \quad (8)$$

After development of programmable logical devices (PLD FPGA) the new capabilities and issues in constructing of digital fault-tolerant systems arose (Sutter *et al*., 2002; Volkoviy and Kharchenko, 2007; Anderson, 2009; Barkalov *et al*., 2013; FPGA &SoC Product, 2015).

One of the leading experts in the field of PLD FPGA development YervantZorian thinks: "Now the main problem of system on a chip repair is development of embedded technologies and methods of the logic repair that occupies no more than 10% of chip area" (Yervant and Dmytris, 2004).

We suggest to retain not initial logic functions, but basis functions, those allows to process initial functions in a longer period of time under specific fault model. We also suggest to provide redundancy of elements bases and compute initial logic functions on residual bases of all elements or subset of elements (Tyurin, 1999).

Classical binary bases (functionally complete systems of Boolean functions) have no such property; therefore we will examine Boolean functions of greater arity by the example of standard model of constant and inverse faults.

We suggest concept of the elements with redundant bases. This is functionally complete tolerant elements (FCT-elements) those retain functional completeness in the terms of Post's theorem (E.L. Post, 1941) under specific fault model (as example, under standard model of constant faults). This allows performing FPGA self-repair and FPGA logic reconstruction after faults

2. METHODS

ANALYSIS OF FUNCTION WITH 3 ARGUMENTS

The common way of finding FCT-function is as follows: we can set some generic vector with yet undefined bit positions a_i, and then we can get all modifications of this vector under specified fault model. Since the bit positions are not defined, we can make up an equation that describes requirements to functional completeness in the terms of Post's theorem. Equation roots, if they exist, describe corresponding FCT-functions.

Let n=3, then to provide constants nonpreserving in case of substitutions it is necessary to put zeroes in bit positions 3, 5, 6 and ones in bit positions 1, 2, 4.

Then we would obtain vector with all known bits, see Figure 3.

Bit position	0	1	2	3	4	5	6	7
Bit value	1	1	1	0	1	0	0	0

Fig 3. Vector of estimated FCT-function for n=3

This function is nonmonotonic, non-affine, but self-dual. This is N23-function. Corresponding Karnaugh map is shown on Figure 4.

x_3 \ x_1x_2	00	01	11	10
0	1	1	0	1
1	1	0	0	0

Fig 4. Karnaugh map for f_{23}

After minimization we have:

$$f_{23} = \overline{x_1}\,\overline{x_2} \vee \overline{x_1}\,\overline{x_3} \vee \overline{x_2}\,\overline{x_3} \quad (9)$$

This is nothing but majority function with inversion. Therefore, number of bit positions in Boolean vector for n=3 does not allow us to get non-self-dual function under single constant faults. However, combining this function with any non-self-dual function we can get a functionally complete tolerant basis (Tyurin*et al*., 2013).

OBTAINING FUNCTIONALLY COMPLETE TOLERANT BOOLEAN FUNCTIONS

We suggest method of obtaining functionally complete tolerant Boolean functions (systems, containing only one function), that allow us to avoid reviewing all modifications of Boolean functions; especially since with the growth of amount of variables, amount of Boolean functions also rises sharply. We assume to create generic vector with provided redundancy of required properties in advance.

Let us consider a specific case for n=4. After providing redundancy of constants non-preserving property we will obtain following vector:

$$111a_3\,1a_5\,a_6\,01a_{10}\,0a_{12}\,000, \quad (10)$$

where the variable a denotes the free bits subscript bit number, starting with 0. Clearly, the vector itself and any of its modifications represent non-monotonic functions. Analysis reveals that the same could be said about affine property. Therefore, we must provide non-self-dual property compliance. Let us consider all modifications of generic vector, see Table 1.

Table 1. Analysis of general vector modifications

Number ofexperiment	Variables				Genericvector
	x_4	x_3	x_2	x_1	
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	a_3
4	0	1	0	0	1
5	0	1	0	1	a_5
6	0	1	1	0	a_6
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	a_9
10	1	0	1	0	a_{10}
11	1	0	1	1	0
12	1	1	0	0	a_{12}
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Modifications of generic vector under input faults are shown on Table 2.

Table 2. Modifications of generic vector under input faults

Modificationsunderinputfaults											
x_1^0	x_2^0	x_3^0	x_4^0	x_1^1	x_2^1	x_3^1	x_4^1	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	a_3	a_5	a_9	1	a_3	a_5	a_9
1	1	1	1	a_3	1	a_6	a_{10}	a_3	1	a_6	a_{10}
1	1	a_3	a_3	a_3	a_3	0	0	1	1	0	0
1	1	1	1	a_5	a_6	1	a_{12}	a_5	a_6	1	a_{12}
1	a_5	1	a_5	a_5	0	a_5	0	1	0	1	0
a_6	1	1	a_6	0	a_6	a_6	0	0	1	1	0
a_6	a_5	a_3	0	0	0	0	0	a_6	a_5	a_3	0
1	1	1	1	a_9	a_{10}	a_{12}	1	a_9	a_{10}	a_{12}	1
1	a_9	a_9	1	a_9	0	0	a_9	1	0	0	1
a_{10}	1	a_{10}	1	0	a_{10}	0	a_{10}	0	1	0	1
a_{10}	a_9	0	a_3	0	0	0	0	a_{10}	a_9	0	a_3
a_{12}	a_{12}	1	1	0	0	a_{12}	a_{12}	0	0	1	1
a_{12}	0	a_9	a_5	0	0	0	0	a_{12}	0	a_9	a_5
0	a_{12}	a_{10}	a_6	0	0	0	0	0	a_{12}	a_{10}	a_6
0	0	0	0	0	0	0	0	0	0	0	0

Let us make a logical equation that describes requirements to generic vector and all its modifications to be non-self-dual under single constant faults x_i^{δ} and single inverse faults $\bar{x}_i$:

$$(a_{12} \vee a_{10} \vee a_6)(a_{12} \vee a_9 \vee a_5)(a_3 \vee a_9 \vee a_{10})(a_3 \vee a_5 \vee a_6)$$
$$(\bar{a}_3 \vee \bar{a}_5 \vee \bar{a}_9)(\bar{a}_3 \vee \bar{a}_6 \vee \bar{a}_{10})(\bar{a}_5 \vee \bar{a}_6 \vee \bar{a}_{12})(\bar{a}_9 \vee \bar{a}_{10} \vee \bar{a}_{12})$$
$$(a_3 a_{12} \vee \bar{a}_3 \bar{a}_{12} \vee a_6 a_9 \vee \bar{a}_6 \bar{a}_9 \vee a_5 a_{10} \vee \bar{a}_5 \bar{a}_{10}). \quad (11)$$

After solving this equation by applying laws of Boolean algebra we will get following solutions:

$$a_3 \bar{a}_5 \bar{a}_{10} a_{12} \vee a_3 \bar{a}_6 \bar{a}_9 a_{12} \vee \bar{a}_3 a_5 a_{10} \bar{a}_{12} \vee \bar{a}_3 a_6 a_9 \bar{a}_{12} \vee \bar{a}_5 a_6 a_9 \bar{a}_{10} \vee a_5 \bar{a}_6 \bar{a}_9 a_{10}. \quad (12)$$

These solutions also provide function's nondegeneracy. Then the definition of corresponding Boolean vectors and minimizing of Boolean functions through the use of Karnaugh map is complete we will get following FCT-functions with 4 arguments:

$$f_{4383} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4, \quad (13)$$

$$f_{4447} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_4, \quad (14)$$

$$f_{4895} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4 \vee \bar{x}_3 \bar{x}_2, \quad (15)$$

$$f_{4954} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_4 \vee \bar{x}_3 \bar{x}_2, \quad (16)$$

The rest of 16 functions distinguished only by numbers of variables:

f_{4415}, f_{5407}, f_{5439}, f_{1327}, f_{1391}, f_{1839}, f_{1903}, f_{855}, f_{887}, f_{1879}, f_{1911}, f_{1336}, f_{5423}, f_{5431}, f_{863}, f_{4951}. (17)

Solution checking shows that, for example, all modifications of $f_{4383}(\bar{x}_2 \vee \bar{x}_3 \bar{x}_4, \bar{x}_1 \vee \bar{x}_3 \bar{x}_4, \bar{x}_1 \bar{x}_2 \vee \bar{x}_4, \bar{x}_1 \bar{x}_2 \vee \bar{x}_3)$ for single faults represent functions with 3 arguments f_{31}, $f_{87}$and function f_1 with 2 arguments. Functions f_{31}, f_{87}have functional completeness property and function f_1 is a well known Peirce basis $\bar{x}_3 \bar{x}_4, \bar{x}_1 \bar{x}_2$.

Basis also retains in the case of inverse faults. Basis $f_{4383}$is the most suitable basis for implementation of Boolean function in DNF class.

The first implementation of 2OR-2AND-NOT FCT-element that represents $f_{4383} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4$ (or, what is the same $\overline{(x_1 \vee x_2)(x_3 \vee x_4)}$) as a CMOS-based transistor circuit is shown on Figure 5.

The second implementation of 2OR-2AND-NOT FCT-element that represents $f_{4954} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_4 \vee \bar{x}_3 \bar{x}_2$ (or $(\bar{x}_1 \vee \bar{x}_2)(\bar{x}_3 \vee \bar{x}_4)$) is shown on Figure 6.

Given elements with redundant basis retain functional completeness under fault of one transistor.

However, in the case of faults, functions of these elements are changing. This makes the recovery process more complicated; we should determine which functions retained and then perform reconfiguration according to maximum possible common basis. If reconfiguration is not possible, we should choose maximum subset of elements those have a common basis.

Custom software was used to analyze all functions with 5 arguments. As a result, 1040384 FCT-functions for single constant and inverse faults and 15264 FCT-functions for double faults were detected.

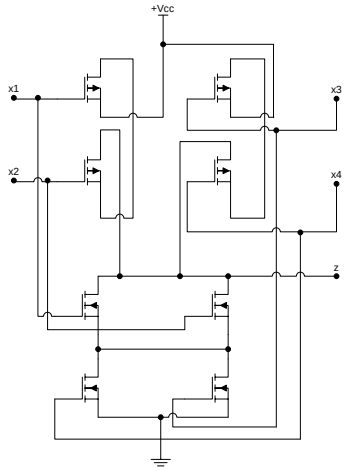

Fig 5. FCT-element represented as a CMOS-based transistor circuit

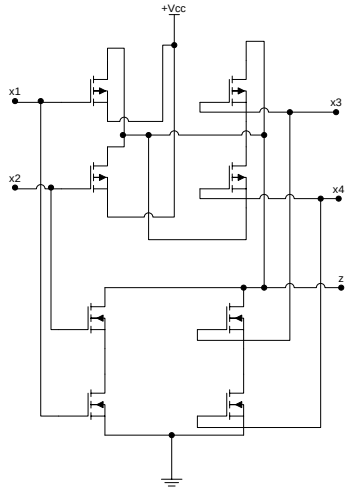

Fig 6. The second possible implementation of FCT-element

SPECIFIC FEATURES OF BRIDGING FAULTS

Let us now analyze an integrated fault model, model that takes into account additional types of faults, like bridging faults: wired AND/OR, dominant, dominant AND/OR, see Table 3.

Taking into account all fault models, we will get a logical equation with 4 variables. The equation describes functional completeness requirements under integrated fault model conditions:

$$(a_6 \vee a_{10} \vee a_{12})(a_5 \vee a_9 \vee a_{12})(a_3 \vee a_9 \vee a_{10})$$
$$(a_3 \vee a_5 \vee a_6)(\bar{a}_3 \vee \bar{a}_5 \vee \bar{a}_9)(\bar{a}_3 \vee \bar{a}_6 \vee \bar{a}_{10})$$
$$(\bar{a}_5 \vee \bar{a}_6 \vee \bar{a}_{12})(\bar{a}_9 \vee \bar{a}_{10} \vee \bar{a}_{12})$$
$$(a_3 a_{12} \vee \bar{a}_3 \bar{a}_{12})(a_6 a_9 \vee \bar{a}_6 \bar{a}_9)(a_5 a_{10} \vee \bar{a}_5 \bar{a}_{10}) = 1 \quad (18)$$

After solving this equation we will get following solutions:

$$(a_3 a_5 \bar{a}_6 \bar{a}_9 a_{10} a_{12}); \quad (19)$$

$$(\bar{a}_3 a_5 a_6 a_9 a_{10} \bar{a}_{12}); \quad (20)$$

$$(\bar{a}_3 a_5 \bar{a}_6 \bar{a}_9 a_{10} \bar{a}_{12}); \quad (21)$$

$$(a_3 \bar{a}_5 a_6 a_9 \bar{a}_{10} a_{12}); \quad (22)$$

$$(a_3 \bar{a}_5 \bar{a}_6 \bar{a}_9 \bar{a}_{10} a_{12}); \quad (23)$$

$$(\bar{a}_3 \bar{a}_5 a_6 a_9 \bar{a}_{10} \bar{a}_{12}). \quad (24)$$

If we compare these solutions with solutions of constant faults model, we will see that bits of solutions

Table 3. Modifications of FCT-vector under bridging of inputs x_2, x_3 and "wired and" bridging

#	x_4	x_3	x_2	x_1	FCT vector	Faults											
						Bridging						x_1 AND x_2	x_1 AND x_3	x_1 AND x_4	x_2 AND x_3	x_2 AND x_4	x_3 AND x_4
						x_2x_3	x_2x_3	x_2x_4	x_2x_4	x_3x_4	x_3x_4						
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	1	a_6	1	a_{10}	1	1	1	1	1	1	1	1	1
3	0	0	1	1	a_3	0	1	1	1	a_3	a_3	a_3	1	1	1	1	a_3
4	0	1	0	0	1	1	a_6	0	a_5	a_{12}	1	1	1	1	1	1	1
5	0	1	0	1	a_5	1	0	a_5	1	0	1	1	a_5	1	1	a_5	1
6	0	1	1	0	a_6	a_6	a_6	0	1	0	1	1	1	a_6	a_6	1	1
7	0	1	1	1	0	0	0	0	a_5	0	a_3	0	0	a_6	0	a_5	a_3
8	1	0	0	0	1	1	1	1	a_{10}	1	a_{12}	1	1	1	1	1	1
9	1	0	0	1	a_9	a_9	a_9	1	0	1	0	1	1	1	a_9	1	1
10	1	0	1	0	a_{10}	0	1	a_{10}	a_{10}	1	0	1	a_{10}	1	1	a_{10}	1
11	1	0	1	1	0	0	a_9	1	0	a_3	0	0	a_{10}	0	1	0	a_3
12	1	1	0	0	a_{12}	1	0	1	0	a_{12}	a_{12}	a_{12}	a_{10}	a_{12}	1	1	a_{12}
13	1	1	0	1	0	a_9	0	a_5	0	0	0	a_{12}	0	0	a_9	a_5	0
14	1	1	1	0	0	0	0	0	0	0	0	a_{12}	a_{10}	a_6	0	0	0
15	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

(13)-(16) correspond to some bits of solutions (19)-(24).

That is, additional limitations, put by bridging model, do not change relevant bit positions, but only extend definition of irrelevant bit positions.

Let us, for example, consider solution $a_3\bar{a}_5\bar{a}_6\bar{a}_9\bar{a}_{10}a_{12}$. Karnaugh map of solution $a_3\bar{a}_5\bar{a}_6\bar{a}_9\bar{a}_{10}a_{12}$ is shown on Figure 7.

x_3x_4 \ x_1x_2	00	01	11	10
00	1	1	1	1
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

Figure 7. Karnaugh map for $a_3\bar{a}_5\bar{a}_6\bar{a}_9\bar{a}_{10}a_{12}$

This Karnaugh map is corresponding to FCT-function $\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4$, that was obtained earlier, but for single constant fault model.

Similarly, let us obtain the rest of FCT-functions, see Table 4.

Table 4. FCT-functions of integrated fault model

#	FCT-function
1	$\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4$
2	$\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_2\bar{x}_3 = (\bar{x}_1 \vee \bar{x}_4)(\bar{x}_2 \vee \bar{x}_3)$
3	$\bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3 = (\bar{x}_1 \vee \bar{x}_2)(\bar{x}_3 \vee \bar{x}_4)$
4	$\bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4$
5	$\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3 = (\bar{x}_1 \vee \bar{x}_3)(\bar{x}_2 \vee \bar{x}_4)$
6	$\bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3$

When constructing tolerance equations, it is necessary to put additional restrictions with regard to dominant variables inversion. In contrast to single constant fault model, new equation contains not disjunction but conjunction of equality conditions of bit positions 3, 12; 6, 9; 5, 10. Integrated fault model has 6 solutions, when single constant model has 20 solutions. All 6 solutions are completely defined with regard to bit positions 3,5,6,9,10,12. Solution $a_3\bar{a}_5\bar{a}_6\bar{a}_9\bar{a}_{10}a_{12}$ is corresponding to FCT-function $\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4$, obtained before for single constant fault model.

3. RESULTS

LOGICAL ELEMENTS THOSE RETAIN BASIS UNDER MULTIPLE FAULTS

For standard fault model FCT-function is represented as combination of different operations NAND, NOR:

$$(\overline{x_1} \vee \overline{x_2})_{,\ldots}\ \overline{x_1 x_2}\ . \tag{25}$$

Suppose k – is a number of faults, redundant basis is tolerant for. For standard bases k=0, for FCT-bases [1…6] k=1.

If k=1, then DNF complexity: 2 conjunctions with 2 variables; CNF complexity: 2 disjunctions with 2 variables. Let us show, that to achieve tolerance for two faults we may use the following bases:

$$\overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee \overline{x_4}\,\overline{x_5}\,\overline{x_6} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ , \tag{26}$$

$$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})(\overline{x_4} \vee \overline{x_5} \vee \overline{x_6})(\overline{x_7} \vee \overline{x_8} \vee \overline{x_9})\ . \tag{27}$$

DNF complexity: 3 conjunctions with 3 variables; CNF complexity: 3 disjunctions with 3 variables.

18 transistors are required for implementation of these elements.

When two constant faults occur, maximum only two conjunctions go down, and one conjunction keeps operating. For instance, if $(x_3 = 1)\,\&\,(x_4 = 1)$, then

$$\overline{x_1}\,\overline{x_2}\,\overline{1} \vee \overline{1}\,\overline{x_5}\,\overline{x_6} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ , \tag{28}$$

or

$$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\ . \tag{29}$$

If $(x_3 = 0)\,\&\,(x_4 = 1)$, then

$$\overline{x_1}\,\overline{x_2}\,\overline{0} \vee \overline{1}\,\overline{x_5}\,\overline{x_6} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ , \tag{30}$$

or

$$\overline{x_1}\,\overline{x_2} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ . \tag{31}$$

If $(x_3 = 0)\,\&\,(x_4 = 0)$, then

$$\overline{x_1}\,\overline{x_2}\,\overline{0} \vee \overline{0}\,\overline{x_5}\,\overline{x_6} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ , \tag{32}$$

or

$$\overline{x_1}\,\overline{x_2} \vee \overline{x_5}\,\overline{x_6} \vee \overline{x_7}\,\overline{x_8}\,\overline{x_9}\ . \tag{33}$$

It is easy to see, that we can take only one of three conjunctions, i.e. under the circumstances of double fault redundant basis reduces by third.

To achieve tolerance to three faults we may use following bases:

$$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4} \vee \overline{x_5}\,\overline{x_6}\,\overline{x_7}\,\overline{x_8} \vee \overline{x_9}\,\overline{x_{10}}\,\overline{x_{11}}\,\overline{x_{12}} \vee \overline{x_{13}}\,\overline{x_{14}}\,\overline{x_{15}}\,\overline{x_{16}}, \quad (34)$$

$$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4})(\overline{x_5} \vee \overline{x_6} \vee \overline{x_7} \vee \overline{x_8})(\overline{x_9} \vee \overline{x_{10}} \vee \overline{x_{11}} \vee \overline{x_{12}})$$
$$(\overline{x_{13}} \vee \overline{x_{14}} \vee \overline{x_{15}} \vee \overline{x_{16}}), \quad (35)$$

$$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4})\overline{(x_5 x_6 x_7 x_8)} \vee (\overline{x_9} \vee \overline{x_{10}} \vee \overline{x_{11}} \vee \overline{x_{12}})\overline{(x_{13} x_{14} x_{15} x_{16})}. \quad (36)$$

DNF complexity: 4 conjunctions with 4 variables; CNF complexity: 4 disjunctions with 4 variables.

However, redundant basis could be reduced in $k+1$ times, and after reducing complexity of basis become equal to:

$$Q^{reduced}{}_k = k+1, k > 0 \quad (37)$$

Required amount of CMOS transistors:

$$S^T{}_k = 2(1+k)^2, k > 0 \quad (38)$$

Let us now consider the possibility of designing basis element that retains logical function under specific fault model.

LOGICAL ELEMENTS THOSE RETAIN FUNCTION UNDER FAULT CONDITIONS

Retention of original logical functions by passive compensation is reached by using so called "quadded logic". At that the following function is implemented:

$$f = f_1 \cdot f_2 \vee f_3 \cdot f_4, \quad (39)$$

or

$$f = (f_1 \vee f_2)(f_3 \vee f_4), \quad (40)$$

where $f_1 = f_2 = f_3 = f_4 = f$.

In case one of four functions has changed, function of entire system retains. For example, if $f_1 = 1$, then

$$f = 1 \cdot f_2 \vee f_3 \cdot f_4 = f \vee f \cdot f = f. \quad (41)$$

If $f_1 = 0$, then

$$f = 0 \cdot f_2 \vee f_3 \cdot f_4 = 0 \vee f \cdot f = f. \quad (42)$$

If the fault occurs, i.e. $f_1 = \overline{f}_1$, then

$$f = \overline{f}_1 \cdot f_2 \vee f_3 \cdot f_4 = \overline{f} \cdot f \vee f \cdot f = 0 \vee f \cdot f = f. \quad (43)$$

Such cases require quadruple redundancy. To retain original function we also may use less costly, but more complicated triple redundancy with the following majority function:

$$f = f_1 \cdot f_2 \vee f_1 \cdot f_3 \vee f_2 \cdot f_3, \quad (44)$$

where $f_1 = f_2 = f_3 = f$.

Therefore, we provide tolerance to faults and failures in one copy of function f.

Choose NOR as a basis function:

$$\overline{x_1}\,\overline{x_2} = \overline{x_1 \vee x_2}. \quad (45)$$

The following expression helps to retain basis function under single constant fault model:

$$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4} \vee \overline{x_5}\,\overline{x_6}\,\overline{x_7}\,\overline{x_8}. \quad (46)$$

It is easy to see, that if

$$\overline{x_{1.1}}\,\overline{x_{2.1}}\,\overline{x_{1.2}}\,\overline{x_{2.2}} \vee \overline{x_{1.3}}\,\overline{x_{2.3}}\,\overline{x_{1.4}}\,\overline{x_{2.4}}, \quad (47)$$

then function $\overline{x_1}\,\overline{x_2}$ does not change if any single constant fault occurs.

For example, if $x_{1.1} = 1$, then left conjunction turns to 0, leaving behind $\overline{x_{1.3}}\,\overline{x_{2.3}}\,\overline{x_{1.4}}\,\overline{x_{2.4}}$, that, clearly, is equal to $\overline{x_1}\,\overline{x_2}$, since $x_{1.3} = x_{1.4} = x_1$; $x_{2.3} = x_{2.4} = x_2$.

If $x_{1.1} = 0$, then we get following expression:

$$\overline{x_{2.1}}\,\overline{x_{1.2}}\,\overline{x_{2.2}} \vee \overline{x_{1.3}}\,\overline{x_{2.3}}\,\overline{x_{1.4}}\,\overline{x_{2.4}}, \quad (48)$$

that is equal to $\overline{x_1}\,\overline{x_2}$, since $x_{1.2} = x_{1.3} = x_{1.4} = x_1$; $x_{2.1} = x_{2.2} = x_{2.3} = x_{2.4} = x_2$.

Also tolerance retains in the case of variable inversion, e.g. if $x_{1.1} = \overline{x}_{1.1}$, then

$$= \overline{x_{1.1}}\,\overline{x_{2.1}}\,\overline{x_{1.2}}\,\overline{x_{2.2}} \vee \overline{x_{1.3}}\,\overline{x_{2.3}}\,\overline{x_{1.4}}\,\overline{x_{2.4}}, \quad (49)$$

and left conjunction turns to 0 either.

Therefore, fault compensation is provided. Also tolerance retains then adjacent circuit lines bridging occurs, or under some multiple faults.

Let us consider link functions for Vcc and Ground buses for FCT+ element based on CMOS transistors, see Figure 8.

$$z_+ = \overline{x_{1.1}}\,\overline{x_{2.1}}\,\overline{x_{1.2}}\,\overline{x_{2.2}} \vee \overline{x_{1.3}}\,\overline{x_{2.3}}\,\overline{x_{1.4}}\,\overline{x_{2.4}}, \quad (50)$$

$$z_- = (x_{1.5} \vee x_{2.5} \vee x_{1.6} \vee x_{2.6})(x_{1.7} \vee x_{2.7} \vee x_{1.8} \vee x_{2.8}). \quad (51)$$

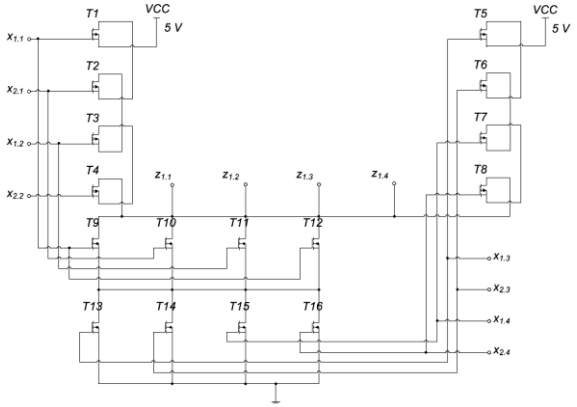

Fig 8. Implementation of FCT+ element based on CMOS-transistors

When transistor representing one of the variables fails, orthogonality of power buses links is not violated:

$$z_+ \oplus z_- = (\bar{x}_{1.1}\bar{x}_{2.1}\bar{x}_{1.2}\bar{x}_{2.2} \vee \bar{x}_{1.3}\bar{x}_{2.3}\bar{x}_{1.4}\bar{x}_{2.4}) \oplus$$
$$\oplus (x_{1.5} \vee x_{2.5} \vee x_{1.6} \vee x_{2.6})(x_{1.7} \vee x_{2.7} \vee x_{1.8} \vee x_{2.8}) = 1. \quad (52)$$

When any single fault occurs, $z_+ = \bar{x}_1\bar{x}_2$ and $z_- = x_1 \vee x_2$.

Let us compare complexity of suggested element with complexity of NOR-element triple redundancy The NOR-element itself consists of 4 transistors, for 3 channels 12 transistors are required.

Majority function:

$$k_1k_2 \vee k_2k_3 \vee k_1k_3. \quad (53)$$

To represent majority function in NOR-basis we will apply double inversion:

$$\overline{\overline{k_1k_2 \vee k_2k_3 \vee k_1k_3}} = \overline{\overline{k_1k_2} \vee \overline{k_2k_3} \vee \overline{k_1k_3}} =$$
$$\overline{\bar{k}_1 \vee \bar{k}_2 \vee \bar{k}_2 \vee \bar{k}_3 \vee \bar{k}_1 \vee \bar{k}_3}. \quad (54)$$

For implementation of this function 52 transistors are required.

Let us measure probability of faultless operation of suggested element. Suppose λ_{fault}– transistor fault rate, $\lambda_{failure}$– transistor failure rate, λ_{inout}– input/output fault rate. When under exponential fault model for simple basis element we have:

$$P_1 = e^{-(4\lambda_{fault}+4\lambda_{failure}+3\lambda_{inout})t}, \quad (55)$$

and for suggested element we have:

$$P_2 = e^{-(16\lambda_{fault}+16\lambda_{failure}+12\lambda_{inout})t}, \quad (56)$$

Let p– generalized probability that one transistor fails, then

$$P_1 = p^4, \quad (57)$$

$$P_2 = p^{16} + 16p^{15}(1-p). \quad (58)$$

Considering probability of one fault either in z_+ or in z_- and 8 cases of simultaneous faults both in z_+ and z_- the above equation turns to:

$$P_2 = p^{16} + 16p^{15}(1-p) + 64p^{14}(1-p)^2. \quad (59)$$

If we consider another types of faults, then we will get 12 options more:

$$P_2 = p^{16} + 16p^{15}(1-p) + 76p^{14}(1-p)^2. \quad (60)$$

Now we will measure the probability of faultless operation ignoring inputs/outputs' fault rate.

To do this, we compare probabilities

$$P_1 = e^{-(4\lambda_{fault}+4\lambda_{failure}+3\lambda_{inout})t} \quad (61)$$

and

$$P_2 = e^{-(16\lambda_{fault}+16\lambda_{failure})t} + 16e^{-(15\lambda_{fault}+15\lambda_{failure})t}(1-e^{-(\lambda_{fault}+\lambda_{failure})t}) +$$
$$+ 76e^{-(14\lambda_{fault}+14\lambda_{failure})t}(1-e^{-(\lambda_{fault}+\lambda_{failure})t})^2. \quad (62)$$

As the Figure 9 shows, due to redundancy FCT+ element has a significant advantage in comparison to NOR-element at larger period of operation.

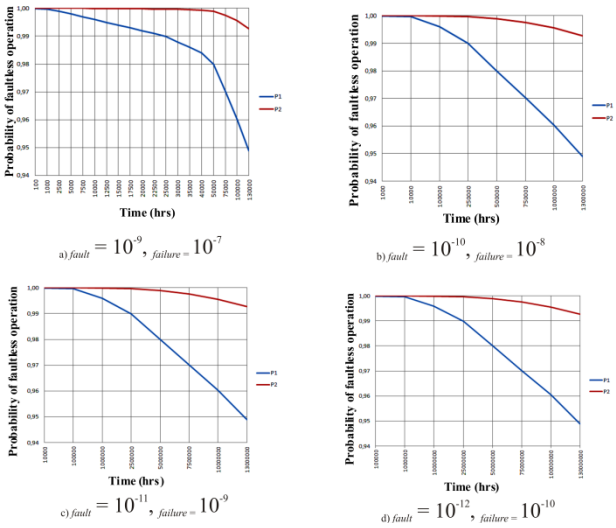

Figure 9. Comparative probabilities of NOR-element faultless operation (P_1) and FCT+ faultless operation (P_2)

FCT+ element could be used as a composite element in the faultless environment and as a fault-tolerant element in the dedicated equipment:

$$f = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_5\bar{x}_6\bar{x}_7\bar{x}_8. \quad (63)$$

To retain basis function 2NOR $(\bar{x}_1 \vee \bar{x}_2)$ under the single fault model the following expression could be used:

$$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)(\bar{x}_5 \vee \bar{x}_6 \vee \bar{x}_7 \vee \bar{x}_8), \quad (64)$$

that is

$$(\bar{x}_{1.1} \vee \bar{x}_{2.1} \vee \bar{x}_{1.2} \vee \bar{x}_{2.2})(\bar{x}_{1.3} \vee \bar{x}_{2.3} \vee \bar{x}_{1.4} \vee \bar{x}_{2.4}) \quad (65)$$

Further development of our approach implies providing fault-tolerance for more complicated FCT-basis $\overline{x_1 x_2} \vee \overline{x_3 x_4}$; for this purpose the following function is required:

$$(\bar{x}_{1.1}\bar{x}_{2.1}\bar{x}_{1.2}\bar{x}_{2.2} \vee \bar{x}_{1.3}\bar{x}_{2.3}\bar{x}_{1.4}\bar{x}_{2.4}) \vee \vee (\bar{x}_{3.1}\bar{x}_{4.1}\bar{x}_{3.2}\bar{x}_{4.2} \vee \bar{x}_{3.3}\bar{x}_{4.3}\bar{x}_{3.4}\bar{x}_{4.4}). \quad (66)$$

Implementation of such element requires 32 CMOS-transistors.

4. DISCUSSION

RETAINING OF LOGICAL FUNCTION UNDER MULTIPLE FAULTS

Suppose k – faults'(failures') multiplicity. We will try to evaluate complexity of multiple faults (failures) compensation. To double faults (failures) of 2NAND basis compensation the following function is used:

$$\overline{x_1 x_2 x_1 x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2 x_1 x_2}. \quad (67)$$

That is, the ninefold redundancy is required. Complexity of expression (67) in letters is equal to 18.

For one fault compensation the fourfold redundancy is required, therefore, compensation complexity is equal to 8. Let us now evaluate complexity of threefold fault compensation. It is obvious, that in this case each conjunction will contain 4 copies of function; overall number of conjunctions is 4 (i.e. greater by 1 than faults' (failures') multiplicity). As a result, it is turns out, that compensation complexity is equal to 32, and redundancy is sixteenfold.

Suppose r – number of letters in implemented function. Then required complexity (S^1) of k-constant faults of function variables, provided that function contains one conjunction consisting of n letters, is equal to:

$$S^1{}_k = (k+1)^2 r. \quad (68)$$

Required complexity (S^2) of k-constant faults of function variables, provided that function contains r conjunctions consisting of n letters each, is equal to

$$S^2{}_k = (k+1)^2 n \cdot r. \quad (69)$$

Required complexity (S^3) of k-constant faults of function variables, provided that function contains r conjunctions consisting of n_r letters each, is equal to

$$S^3{}_k = \sum_{i=1}^{r} (k+1)^2 n_i. \quad (70)$$

Required complexity (S^4) of k constants faults in any (not necessarily functionally complete) system of m functions, provided that each function contains $r_m$conjunctions consisting of $n_r$letters each, is equal to

$$S^4{}_k = \sum_{j=1}^{m} \sum_{i=1}^{r} (k+1)^2 n_i. \quad (71)$$

BRIDGING FAULTS

Let us consider bridging faults. For two variables x_i, $x_j$the following types of bridging are possible:

1) wired AND: $x_i = x_j = x_i \wedge x_j$;
2) wired OR: $x_i = x_j = x_i \vee x_j$;
3) dominant-substitution: $x_i$is substituted by x_j and vice versa;
4) dominant AND: both x_i and x_j are substituted by $x_i \wedge x_j$;
5) dominant OR: both x_i and x_j are substituted by $x_i \vee x_j$

We will show that bridging of any two variables in one conjunction does not change the function. Represent (47) as:

$$\overline{x_1 x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2}. \quad (72)$$

Apparently, bridging of any two variables in one conjunction does not lead to emerging of new variable or constant and does not change the logical value of conjunction. For example, then the dominant OR of x_2 in first conjunction occurs, expression (72) turns into:

$$\overline{x_1(x_2 \vee x_1)x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2}, \quad (73)$$

that is equivalent to (72).

Bridging of any two variables in different conjunctions does not lead to emerging of new variable or constant and does not change the logical value of conjunction either. For example, then the dominant OR of x_1 in first conjunction and dominant OR of x_2 in second conjunction occurs, expression (72) turns into:

$$\overline{(x_1 \vee x_2)x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 (x_1 \vee x_2)}, \quad (74)$$

that is equivalent to (72) either.

However, then the triple bridging occurs, and x_1 is dominating over x_2, expression (72) turns into:

$$\overline{x_2 x_2 x_2 x_2} \vee \overline{x_1 x_2 x_1 x_2}, \quad (75)$$

and that is not equal to initial expression (72).

In such cases we should increase conjunctions' length and their number, just as we do in the case of double fault compensation.

For FCT-function with 4 variables, when first copy of x_1 and the first copy of x_3 are bridged, we will get:

$$\overline{x_1 x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2} \vee \overline{x_1 x_4 x_3 x_4} \vee \overline{x_3 x_4 x_3 x_4}, \quad (76)$$

i.e., an extra variable is added in the third conjunction.

However, this extra variable would be "absorbed" by the fourth conjunction:

$$\overline{x_1 x_2 x_1 x_2} \vee \overline{x_1 x_2 x_1 x_2} \vee \overline{x_3 x_4 x_3 x_4}. \quad (77)$$

So, if triple bridging is allowed, then we have to increase conjunctions' length and their number.

Therefore, costs associated with bridging compensation are equal to costs associated with *k*-constant faults (Kamenskih and Tyurin, 2015).

5. CONCLUSION

Suggested elements with redundant bases implement a sort of "internal" redundancy, and they could be used in designing of high reliable digital circuits tolerant both to faults and failures. Elements, that retain basis function under specific fault model and that provide operability on bases' subset, are recommended to use if the outage is allowed. Then so called active fault tolerance is provided due to external diagnostic equipment; i.e. residual bases are later used in FPGA reconfiguration. This is the most inexpensive option. If the outage is not allowed, then we suggest to implement passive-active fault tolerance based on structure redundancy by recovery of inoperable elements in every channel.

The most valuableoption is passive-active fault tolerance based on elements those retain implemented logical function under specific fault model.

Besides, suggested approach contributes to yield rate increasing in the integrated microcircuit manufacturing.

The results create the preconditions for the creation of high-reliability computers, in which the workers, monitoring and recovery processes are a single entity that can operate without maintenance and repair throughout the life cycle. It is necessary for the control equipment space, aviation, military complex, hazardous process, nuclear power plants, in agriculture - for advanced intelligent computer systems in the agricultural, livestock in areas with difficult climatic conditions, far from service centers, with the possible low-skilled staff.

Analysis of trends in science and technology shows that intelligent digital equipment of the new information civilization will be capable of self-healing and adaptation to failure and damage, for example, by disabling the affected areas and implementing the required functions on the remaining number of items with a possible slowdown in the rate of admissible.

References

[1] Anderson, J. Low-Power Programmable FPGA Routing Circuitry / Anderson, J. // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. -Vol. 17. - No. 8. - 2009. - P. 1048-1060.

[2] AvižienisA. Fault-Tolerance: Thesurvivalattributeofdigitalsystem / A. Avižienis // Proc. oftheIEEE. – 1978. –Vol. 66, No 10. – P. 1109-1125.

[3] AvizienisA., LaprieJ.-C. Dependable computing: From concepts to application // IEEE Trans. on Computers. – 1986. –No 74(5). – P.629-638.

[4] AvižienisA. Basicconceptsandtaxonomyofdependableandsecurecomputing / AvizienisA., Laprie J.-C., Randell B., Landwehr C.// IEEE Transactions on Dependable and Secure Computing, vol. 1, No 1, 2004. - P. 11-33.

[5] Barkalov, A. Hardware Reduction in FPGA-based Moore FSM / Barkalov, A., Titarenko, L., Malcheva, R., Soldatov, K.//Journal of Circuits, Systems and Computers, Vol 22, 3, 2013. - P. 1-20.

[6] Brown, M. (1980). "Bounds, Inequalities, and Monotonicity Properties for Some Specialized Renewal Processes".*The Annals of Probability*8(2): 227.doi:10.1214/aop/1176994773.

[7] Brown, M. (1981). "Further Monotonicity Properties for Specialized Renewal Processes".*The Annals of Probability*9(5): 891.doi:10.1214/aop/1176994317.

[8] E.L. Post, The two-valued iterative systems of mathematical logic, Annals of Mathematics studies, no. 5, Princeton University Press, Princeton 1941, 122 pp.

[9] Finkelstein, Maxim (2008). "Introduction".*Failure Rate Modelling for Reliability and Risk*. Springer Series in Reliability Engineering. pp. 1–84.doi:10.1007/978-1-84800-986-8_1.

[10] FPGA &SoC Product (2015, April 15). Retrieved April 15, 2015, from http://www.microsemi.com/products/fpga-soc/fpga-and-soc/.

[11] J. Von Neumann. Probabilistic logic and the synthesis of reliable organisms from unreliable components. Automata Studies, C. Shannon and J. McCarthy (eds). Princeton University Press, 1956, pp. 43-98.

[12] Kamenskih, A.N.,Tyurin, S.F. Features that provide fault tolerance of self-synchronizing circuits //Russian Electrical Engineering. –2015.P.672-682.

[13] SambhuNath Pradhan. Low Power Finite State Machine Synthesis Using Power-Gating / SambhuNath Pradhan, M. Tilak Kumar, Santanu Chattopadhyay // INTEGRATION, the VLSI journal. - 44 (2011). - 2011. - P.175–184.

[14] Shanthikumar, J. G. (1988). "DFR Property of First-Passage Times and its Preservation Under Geometric Compounding".*The Annals of Probability*16:397–406.doi:10.1214/aop/1176991910.

[15] Sutter, Gustavo. FSM Decomposition for Low Power in FPGA / Sutter, Gustavo, Todorovich, Elias, Lopez-Buedo, Boemo, Sergio Eduardo // Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications. - 2002. - P. 350-359.

[16] Tyurin S.F. Retention of functional completeness of Boolean functions under "failures" of the arguments. Automation and Remote Control. – 1999. Vol.60. No 9 part 2. pp. 1360-1367.

[17] Tyurin, S.F., Grekov, A.V., Gromov, O.A. The principle of recovery logic FPGA for critical applications by adapting to failures of logic elements (2013) World Applied Sciences Journal 26 (3) pp. 328 - 332 doi: 10.5829/ idosi.wasj. 2013.26.03. 13474.

[18] Volkoviy, S. Improving of Technical Characteristics of Systems-on-Programmable Chips Using Internal Circuit Diversity / Volkoviy, S., Kharchenko, V. // Bulletin of

Khmelnitsky National University, Technical Science. - No 2. - 2007. - P. 153-155.

[19] Yervant Z. Gest editors' introduction: Design for yield and reliability/ Z. Yervant, G. Dmytris // IEEE Design & Test of Computers. - May-June 2004.-P. 177-182.

[20] Yu, Y. (2011). "Concave renewal functions do not imply DFR interrenewal times".*Journal of Applied Probability*48(2): 583.doi:10.1239/jap/1308662647.