Structural Redundancy and Design Space Exploration Method for the Hardware Components with Fault Mitigation Design

Valentin Rozanov, Yuriy Sheynin, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation
{suvorova, sheynin}@aanet.ru, valentin.rozanov@guap.ru

***Abstract*—Fault mitigation for modern embedded systems is a necessary feature due to accelerating aging and manufacturing defects, which diagnosis during the chip testing at fabric is impossible. In addition, different ways of system using may need different degree of fault protection that need to be implemented. Another parameter of embedded system – area. It is one of most critical parameters for SoC in embedded systems and is strongly constrained. Increasing fault protection leads to growing of the SoC's area. In this situation, it is necessary to know how strong the fault mitigation is and how it effects on the area. We propose the method for development of hardware components that can help to evaluate project from point of area constraints and fault probability requirements.**

I. INTRODUCTION

Building on-board network is impossible without using network controllers.

In the development process, the transport layer controller is an IP block, which is further used as a part of the SoC performing various functions on the on-board network. Transport layer controller's IP block, being more complex in structure and function, than IP blocks of the lower layers of the data transfer protocol, have a higher probability of failure than the Besides that using of thin design rules for SoC allow to place a lot of different components on one chip. Therefore, the functionality of embedded systems grows dramatically. However, using of thin design rules is accompanied with accelerated aging and manufacturing defects that can not be diagnosed during the chip testing at the fabric [1]. Therefore manufactured by thin design rules SoC should include fault mitigation mechanisms [2], [3], [4]. Thus, the task of mitigating the faults that occur in such an IP block is very urgent.

Any transport controller includes elements such as memory (for storing configuration, buffering data), a state machine (one or more that determine the state of the connection, the data being transmitted), and logic that ensures the functioning of the memory blocks and the state machine.

Controllers used in on-board computer networks are exposed to charged particles, which can disrupt the normal operation of the controller, or the information stored in it. Damage to a function block is determined by the location of the charged particle in the controller's area and location of the functional block on this area.

Two variants may be considered when an fault occurs in state machine:

1) Soft fault – as a result of a value change at the FSM input (as a result of the action of charged particles on the register or memory from which parameters of FSM are read), the transition to the state erroneous for the current operating stet of the FSM will occur. In this case, there may be a loss of transmitting data, or a disconnection of the controller. However, the state machine component will work, and can recover after the failure.
2) Hard fault – failure of FSM because of physical failure of registers or communication lines between logical components. In this case, it is impossible to restore the controller's efficiency.

In the controller operating conditions in the on-board computer networks, it is necessary to provide the possibility of detecting and mitigating of such faults. For convenience, we will call the complex of detection and mitigating faults "protection". Implementing a hardware controller it is necessary to take into account the area that controller will occupy in production. The use of "protection" tools inevitably increases the area occupied by the controller. To regulate the size of the occupied area, it is possible to use various combinations of detection and mitigating mechanisms, as well as the complexity of these algorithms. Therefore it is necessary to be able to analyze possible options for using "protection" in terms of achieving the required probability of failure (or uptime), and the area that will be occupied by a functional component with a "protection" mechanism.

In this article, only a part of the "protection" will be considered - the mechanism for fault mitigation. This is due to the fact that the detection mechanism is a large area that requires a separate considering, and is beyond the scope of the work being presented.

In the proposed research, the estimation of the parrying mechanism was evaluated according to the following parameters:

1) Number of steps to fail – number of steps after which probability to stay in "failed" state is more than specified

2) Area of the functioning block with mitigation mechanism

Following values were the variable parameters were the:

1) Construction of the functional component
2) Number of reserved elements
3) Size of input and output data vectors of functional component

The research objective is to construct a solution space for functional component that perform the same functionality, having the same size of input and output data, but differing the way they are implemented.

To solve the problem we propose a method of partial redundancy selection that is based on design space exploration (DSE). Based on DSE methods are widely used for modern SoC development [5], [6], [7], [8]. The design space exploration of NoCs is commonly formulated as a constrained optimization problem [9]. This approach is used for different tasks (such as buffer size selection, arbitration rules selection and many others) that have high computation complexity.

Solving the problem is divided into 5 main parts.

The first part (Section 2) describes options for constructing a functional block, and a mechanism for fault mitigation. Options for full and partial redundancy were chosen for the analysis. The second part (Section 3) - defines approaches to the method of constructing functional blocks with a mitigating mechanism based on the DSE. The third part (Section 4) considers the method of DSE applicable to the functional blocks selected for consideration. The fourth part (Section 5) represents an estimate of the probability of failure of a functional block of different architectures with a fault mitigation mechanism. The fifth part (Section 6) presents the results of estimating the failure probabilities and the effect of the parry method used on the area of the functional block for various methods of its construction.

II. FUNCTIONAL BLAOCKS' CONSTRUCTION WITH FAULT MITIGATION MECHANISM

There are various approaches of building component partial redundancy. We consider two most common ways of partial redundancy as use cases: the whole component redundancy and sliding redundancy of subcomponents [1].

Some spare components are included into the system, when the main component functionality is critical for the SoC. Quantity of spare components is equal to number of faults, against which the system should be tolerant. Using one spare component when the main component is failed allows supporting the SoC functionality without degradation. Fig. 1 describes the situation with one spare component in the scheme. If faults are detected, control signal is applied, that changes data flow direction from main component to spare component in input and output MUX.

This approach essentially increases the SoC hardware cost (area), while the strong area constraints are typical for many embedded systems. Therefore, it is often impossible to provide redundancy for many components, which functionality is critical for the SoC. It significantly reduces operating parameters of the SoC when this approach is used, and essentially limits its scope.

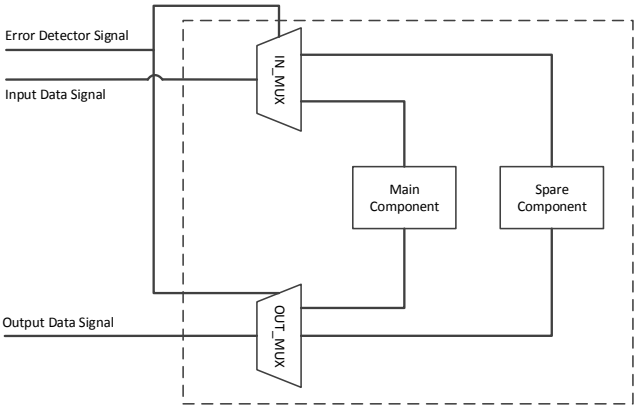

Fig. 1. The example of full redundant structure

Also we consider another way of partial redundancy in this paper. Realization of this approach would require less area. It is based on decomposition of a basic component onto subcomponents that are self-similar to the basic component.

For most components whose inputs and outputs are bit vectors decomposition on self-similar sub-components may be used. These self-similar components will can process parts of input/output vectors. So we can say that one component that have N input/output bits may be divided into M sub-components that will have N/M input/output bits.

However, implementation of self-similar sub-components involves additional overheads (with increasing of area and timing constraints). In NoC can be identified quite a number of types of components overhead for implementation of which as a group of components are not big.

In case of decomposition there is an opportunity to use spare sub-component instead of spare component (Fig 1). For protection the component against one failure, we don't need to duplicate full component. Quantity of spare subcomponents should be equal to quantity of mitigated failures.

Fig. 2 describes including spare sub-component in the decomposed component. IN-MUX (as on Fig. 1) divides input data between sub-components. Spare MUX receives data from both lines and switches which data spare sub-component need to operate. In this situation, spare sub-component can operate data instead of sub-component 1 or 2. OUT-MUX combine data from different sub-components, and may use data from spare sub-component if it is needed. All MUXes are applied by one line from Fault Detector. This method of redundancy is called sliding redundancy. The dashed line indicates the area that is considered in this article. This area covers subcomponents and multiplexers that are used to switch lines between main and spare subcomponents. This article does not include exploration of the source of the signal switching the circuit from the main subcomponents to the spare ones, i.e. the article does not consider the issue of detecting the occurrence of faults.

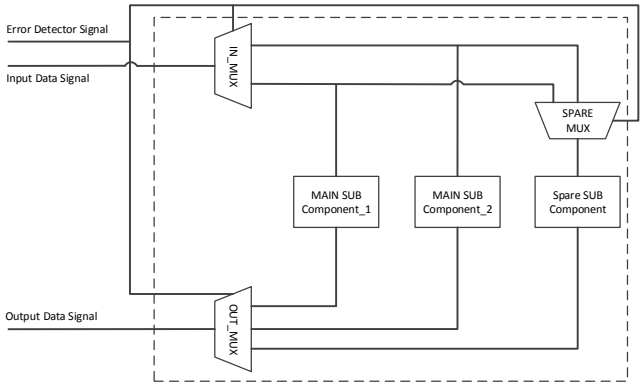

Fig. 2. The partial redundancy scheme on subcomponent layer

Using spare MUX is that overheads that differs full redundancy and sliding redundancy. But in sliding redundancy we need much less area for spare sub-component than spare component in full redundant scheme.

Using of these approaches allow to implement components that resist to equal quantity of faults. But there construction logic of functioning differs.

The multiplexers, used for including spare components are bottleneck for both schemes. The considered approaches do not allow to mitigate faults in these multiplexers. Quantity of multiplexers and its area are various for different approaches, therefore fault probability will be different.

In this paper we propose the components partial redundancy method based on design space exploration. This method allow to choose partial redundancy way correspondingly the area constraints and required fault probability.

For the scheme with full redundancy, base component is main component that is reduced with spare component. In sliding redundancy, scheme base component is one main sub-component, because all other sub components are the same with the base.

Different approaches of components partial redundancy are used for SoCs [10], [11], [12], [13]. All approaches of partial redundancy lead to increasing of SoC's area. Area of spare components and the fault probability could be various for different approaches and depend on way of spare components placing. The required fault probability depends on planned embedded system lifetime.

In many cases the smallest area overheads lead to smallest fault probability. However, dependency between these parameters is very complex. It is determined by scheme of spare components integration into the system, by the size (area) of additional multiplexers that are used. These multiplexers themselves do not have redundancy, fault mitigation for them is not implemented, thus they could decrease achievable fault probability.

Therefore the operating parameters of a developed embedded system strongly depends on selected approach for partial redundancy in it.

The N-dimensional design space is formed in the frame of this approach. Number of dimensions is equal to quantity of system parameters, for which values should be specified or constraint in the system design.

III. Method of the Functional Blocks' Construction Based on the DSE

Design space in the considering situation may be determined by three parameters:

- area that is needed to place the logic;
- time (or number of steps if discrete model is considered) to reach maximum permissible value of the probability to fail;
- number of the sub-components that are used to construct the component.

To define each of the parameters it is necessary to go through the algorithm that may be described with following steps:

1) Development of the basic structure (without spare components) and development of the structure with spare components for every considered partial redundancy way.
2) Evaluation of area for every component included into the structure.
3) Evaluation of the area for the basic structure.
4) Evaluation of the area for the structure with spare components.
5) Evaluation of the area overheads that arises due spare components – obtaining the coordinate of the point on the area overheads axis.
6) Development of the Markov net for the structure with spare components.
7) Evaluation of probabilities to fail for the elements of Markov net correspondingly its areas and types.
8) Evaluation of fault probability - obtaining the coordinate of the point on the fault probability axis.

IV. Determination of the Component Area on the Basis of the Areas of the Basic Component and Overheads

We introduce following notations:

Sb – the area of base component

Sc – the area of one subcomponent (self-similar to base component) for second way of partial redundancy

$$Sc=\frac{Sb}{N},$$

where N – quantity of subcomponents in base component.

Sr – the area of scheme with partial redundancy, Sr1 – the area when first way is used, Sr2 – the area when second way is used.

$$Sr1 = (K+1) * Sb + Sm \quad (1)$$

$$Sr2 = N * Sc + (K+1) * Sc + Sm2 = Sb + (K+1) * Sc + Sm2 \quad (2)$$

$$Sh1 = Sr1 - Sb = K * Sb + Sm \quad (3)$$

$$Sh2 = Sr2 - Sb = (K+1) * Sc + Sm2 \quad (4)$$

where

K – redundancy multiplicity

Sm1 – the area of multiplexers

Sh – the area overheads,

Sh1 – the area overheads when the first way is used,

Sh2 – the area overheads when the second way is used

V. MARKOV CHAIN FOR THE COMPONENTS WITH FAULT MITIGATION

To estimate the probability of the occurrence of hard error and the failure of a component, it was decided to consider the state of the component in the form of a Markov chain. This consideration allows you to evaluate possible scenarios of events during the work of the component.

We introduce the following notation for states:

W – fully operational component, all subcomponents are functioning

P – fault of one of the main subcomponents
R – fault of the spare subcomponent
F – fully unworkable state of component

Markov chain for the component with fault mitigating mechanism may look like on Fig. 3.

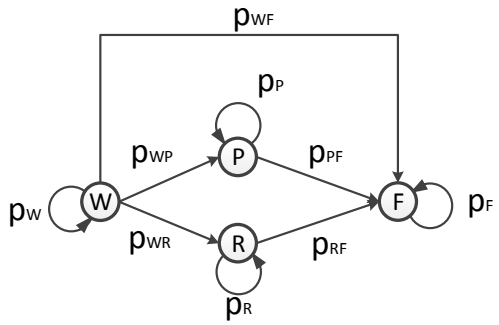

Fig 3. Markov chain for the component with one fault mitigation

The resulting chain is absorbing. This is due to the fact that over time, any device goes into a inoperative state. Transitions between states are caused by the occurrence of certain events:

1) The transition W¬> P occurs when the main subcomponent or its multiplexer fails. In this case, only one of the subcomponents is denied, the rest are in a working state. The redundant subcomponent for this transition is considered workable
2) The transition W¬> R occurs in the event of failure of the spare subcomponent. The main subcomponents are considered to be workable at the same time.
3) The P>F transition occurs in the same way as the W->F transition; however, for P →F, one subcomponent (and more), including the back-up one, suffices.
4) The transition R¬> F occurs in the event of the failure of one (or more) subcomponent. In this case, the backup subcomponent has already been denied.
5) The transition W¬> F occurs when two or more subcomponents fail. In this case, it can be two or more main subcomponents, or a combination of a spare subcomponent and the main subcomponent (one or more), or the failure of all the subcomponents at once

However, this graph can be generalized, in view of the fact that the transitions W¬> P, W¬> R, and P¬> F, R¬> F can be considered together as a logical "OR" construction. Thus, the graph will look like the one shown in Fig. 4.

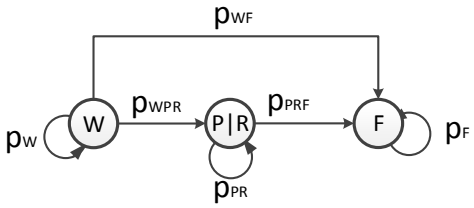

Fig 4. Markov chain for the component with one fault mitigation. Generalized view

In this form, the graph is suitable for any number of sub-components and describes mitigating one hard fault.

The following notation is used in the graph:

p_W – transferring probability to stay in W-state (work without failures)

p_{WF} – transferring probability to move from state W to F-state, where F – state when component is completely non-working

p_{WPR} – transferring probability to move from state W to state P|R,

p_{PRF} – transferring probability to move from state P|R to F-state (system failed).

p_{PR} – transferring probability to stay in PR-state

p_F – transferring probability to stay in F-state. As F is the finish state, this probability is equal 1

Bsed on description of transfers in received generalized graph formulas were derived. These formulas describes transferring probability in graph on Fig. 5. Formulas form is justified by the fact that probability to fail of main subcomponent and spare subcomponent are different.

Formula 5 describes the transition probability P_{WPR}. This probability is the sum of the probabilities of a hard fault in the spare subcomponent when the mains are in good condition, or in one of the main subcomponents with a good condition of spare one.

$$P_{WPR} = (1 - P_R) \cdot C_N^1 \cdot P \cdot (1 - P)^{N-1} + \\ + P_R \cdot (1 - P)^N \quad (5)$$

where:

P – main subcomponent fail probability

P_R – reserve subcomponent fail probability

N – number of states in the scheme (excluding W и F states)

Equation 6 describes probability to transfer wrom W-state

$$P_{WF} = (1 - P_R) \cdot \sum_{n=2}^{N} C_N^n \cdot P^n \cdot (1 - P)^{N-n} + \\ + P_R \cdot \sum_{n=1}^{N} C_N^n \cdot P^{N-n} \cdot (1 - P)^n \quad (6)$$

to F-state.

where:

n – minimal number of failed elements

Formula 7 describes the probability of transition from the state P|R to the state F. This situation occurs when one main subcomponent fails (or more) when the spare sub-component is not failed, or if the spare subcomponent fails when main one are not failed, or if the spare subcomponent and one (and more) of the main subcomponent fails.

$$P_{PRF} = \sum_{n=1}^{N} C_N^n \cdot P^n \cdot (1 - P)^{N-n} + \\ + (1 - P_R) \cdot \sum_{n=1}^{M} C_M^n \cdot P^n \cdot (1 - P)^{M-n} + \\ + P_R \cdot \sum_{n=0}^{M} C_M^n \cdot P^{M-n} \cdot (1 - P)^n \quad (7)$$

where:

M – number of working subcomponents, and M=N-1

Probability to appear in states W, P|R will be defined by

$$P_W = 1 - P_{WPR} \quad (8)$$

$$P_{PR} = 1 - P_{PRF} \quad (9)$$

equations (8-9)

In this case probability to stay in state F will be equal 1, because the chain is absorbing, and the state F is absorbing one.

VI. Results of DSE method using

The proposed construction of the redundancy options have been implemented and synthesized using Cadence RTL Compiler 16.1. As the result of compiler work areas of components were obtained. Initially, the basic circuit elements were synthesized –with 2/4/8/16/32/64 bit width of input/output vectors. Table I contains areas and fault probability of the component with different width of the vector and different area correspondingly.

TABLE I. Synthesis of base components

Input vector Bit Width	Total Area	Probability to get a fail on this area
2	1330	$2 \cdot 10^{-6}$
4	2510	$4 \cdot 10^{-6}$
8	5600	$9 \cdot 10^{-6}$
16	11289	$1.8 \cdot 10^{-5}$
32	25034	$8 \cdot 10^{-5}$
64	61733	$1 \cdot 10^{-3}$

Tables II and III contains information about area of components constructed with subcomponents. In Table II – without redundancy, in Table III – with partial redundancy. As for partial redundancy multiplexor is needed area of multiplexor is also presented.

TABLE II. Synthesis of different combinations of sub-components without redundancy

Input vector Bit Width	Number of sub-components	IN/OUT vector Width	Base Area	Total Area
32	4	8	5600	22400
32	2	16	11289	22578
64	8	8	5600	44800
64	4	16	11289	45156

TABLE III. Synthesis of different combinations of sub-components with redundancy

Input vector Bit Width	Number of sub-components	IN/OUT vector Width	Base Area	MUX Area	Total Area
32	5	8	5600	3400	45000
32	3	16	11289	4950	48717
64	9	8	5600	4100	87300
64	5	16	11289	6200	87445

Table IV contains average MUX area and probability to fail that depends on the area of MUX. This probability and value from Table I forms P_R value, because spare subcomponent is considered together with MUX.

TABLE IV. SYNTHESIS OF DIFFERENT COMBINATIONS OF SUB-COMPONENTS WITH REDUNDANCY

Input vector Bit Width	Average MUX Area	Probability to get a fail on this area
2	620	$1 \cdot 10^{-6}$
4	970	$1.5 \cdot 10^{-6}$
8	1120	$1.8 \cdot 10^{-6}$
16	2150	$3.4 \cdot 10^{-6}$
32	4175	$6.7 \cdot 10^{-6}$
64	8444	$1.3 \cdot 10^{-5}$

Based on the data from Tables I-III, the following parameters necessary for building the solution space can defined:

- Area of base subcomponent
- Overhead area
- Probability to fail depending on the area (subcomponents' or MUXs')

Let's build the solution spaces for the described components, using the formulas from Section 4 to determine the areas, and the formulas from Section 5 to determine the transition probabilities and the number of steps to failure. Calculations will be carried out for components having 32 and 64 bits of input / output vector. Each of them can be made up of 2, 4 or 8 subcomponents having 16, 8, 4, 2 bits at the input/output of the each subcomponent.

Table 5 shows the results of calculating the number of steps to failure for components of different architectures (size of input/output vector and the number of subcomponents). The probability value is calculated based on a probability of failure of the area unit. Thus, the formula for calculating the probability of failure of the main subcomponent or spare

$$P = p \cdot S \tag{10}$$

subcomponent can be calculated by formula (10)
where:

P – probability to fail main or spare sub-component
p – base value of the probability to fail
S – area of the sub-component

Base probability to fail value was used - $p=1.6 \cdot 10^{-9}$

TABLE V TOTAL AREA OF THE COMPONENTS CONSTRUCTED WITH DIFFERENT NUMBER OF SUB-COMPONENTS

IN/OUT Data Width	Combination	Steps To Fail	Total Area
32	2 × 16	102972	22578
	4 × 8	119340	22400
	8 × 4	145205	22170
	16 × 2	153514	22280
64	2 × 32	23890	45320
	4 × 16	59785	45156
	8 × 8	65643	45098
	16 × 4	77257	45265

On the basis of the calculations carried out, graphs can be constructed (Fig. 5 and Fig. 6).

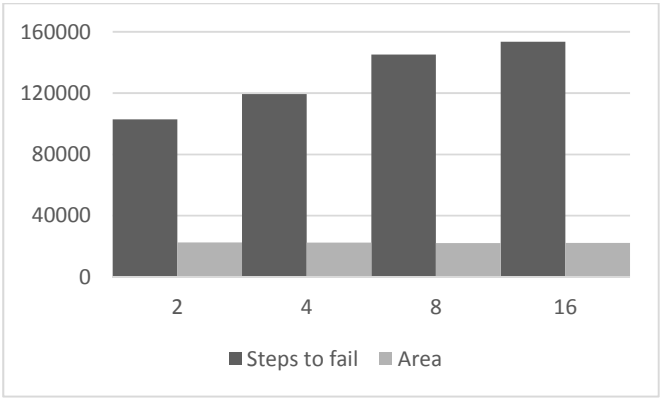

Fig 5. Solution space for component with 32 bits in/out vector width

On the graphs there are numbers of sub-components used in the component (2, 4, 8, 16).

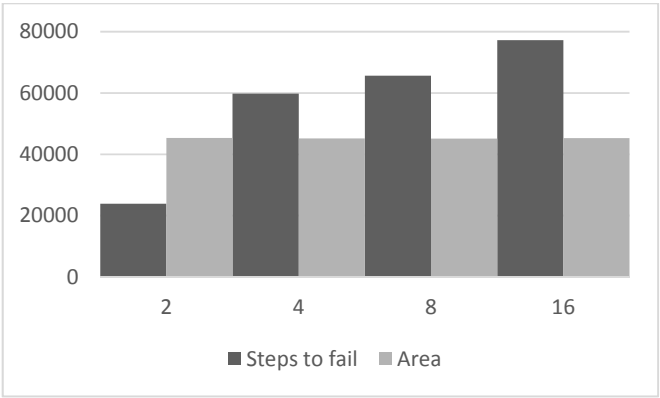

Fig 6. Solution space for component with 64 bits in/out vector width

Graphs on Fig 5 and 6 shows increasing of the steps to fail with the increasing of complexity of internal component structure. The area of the component is very similar. However, as it may be seen from Table V it is different for different variants of internal construction of the component. More clearly it shows the graph on Fig 7.

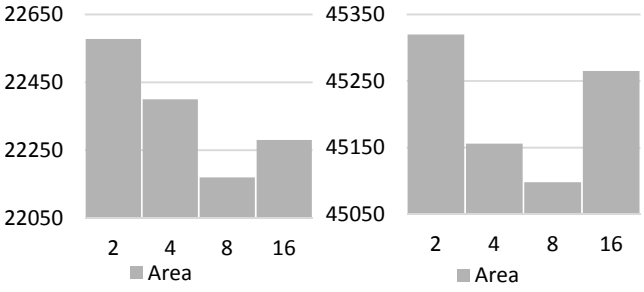

Fig 7. Areas of the components with different number of subcomponents

Graph on Fig 8 shows increasing of probability to appear in state F of Markov chain.

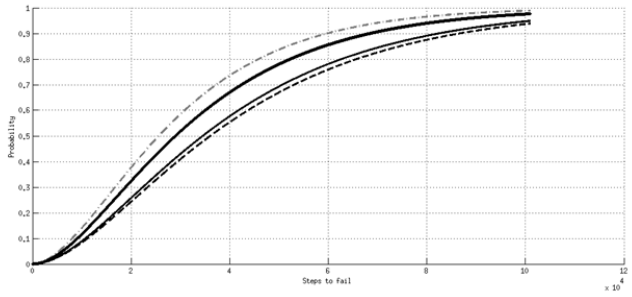

Fig 8. Increasing of probability to appear in state F

CONCLUSION

The article describes an algorithm, that makes possible using of DSE method in SoC components and functional blocks design. The proposed DSE method is based on three parameters: the area of the functional block, its internal architecture, and the probability of a hard error appearing in the subcomponents of the functional block.

The proposed method involves the use of Markov chains, which, in turn, require a parameter - the probability of a transition. To determine the transition probabilities, formulas were derived (Section 5). These formulas can be used for functional blocks of a similar architecture.

The article gives an example of using DSE method for a component with different architectures, but identical functional. Based on the results of the calculations that are necessary to use the method, graphs were constructed that reflect the solution space for the selected example.

During this research important remark was made - changing complexity of the internal structure of the component causes changing of the probability and area. It is necessary to search how probability changes depends on changing of complexity.

Also, during the work on this research, was mentioned that synthesis of redundant components need to be carried with special parameters, because of using different optimization algorithms in synthesis tools. This optimizations may lead to disappearing necessary redundancy on the gate level and transistor level in favor of area minimizing.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS), 2013.

[2] Armin Runge, “FaF NoC: a Fault-tolerant and Buerless Network-on-chip”, Procedia Computer Science, vol. 56, 2015, pp. 397–402.

[3] Erica Cota, Alexandre de Morais Amory and Marcelo Soares Lubaszewski, Reliability, Availability and Serviceability of Networks-on-Chip. Springer, 2012.

[4] Pooria M. Yaghini, Ashkan Eghbal, Hossein Pedram and Hamid Reza Zarandi, “Investigation of transient fault effects in synchronous and asynchronous Network on Chip router”, Journal of Systems Architecture, vol. 57, issue 1, Jan. 2011, pp. 61–68.

[5] System Level Approach to NoC Design Space Exploration. R. K. Jena. International Journal of Information and Electronics Engineering, Vol. 2, No. 2, March 2012 / 5 p

[6] R. K. Jena and G. K. Sharma, “A Multi-Objective Evolutionary Algorithm Based Optimization Model for Network-on-Chip Synthesis,” in Proc. of 4th International conference on IT: New Generation, April, 2-4, Las Vegas, Nevada, USA,2007 pp. 977-983.

[7] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast and Z. Wang, A NoC Tra_c Suite Based on Real Applications, 2011 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pages 66-71, ISSN 2159-3469, July 2011.

[8] Qualcomm, Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age, Qualcomm white paper, October 2011, https://developer.qualcomm.com/download/qusnapdragons4whitepaperfnlrev6.pdf, Retrieved 18 April 2014.

[9] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular NoC architectures,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 24(4), 2005

[10] Rozanov V. Suvorova E. Approaches to the SoC IP-blocks’ Design with Errors’ Mitigation, Proceedings of the FRUCT’19, 2016 pp 196-202

[11] Y.C. Chang, C.T. Chiu, S.Y. Lin and C.K. Liu, “On the design and analysis of fault tolerant NoC architecture using spare routers”, in Proceedings of the Asia and South pacific design automation conference (ASPDAC), 2011, pp. 431–436.

[12] Yu Ren , Leibo Liu , Shouyi Yin, Jie, Qinghua Wu and Shaojun Wei, “A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration”, Journal of Systems Architecture, vol. 59, 2013, pp. 482–491.

[13] C. Liu, L. Zhang, Y. Han and X. Li, “A resilient on-chip router design through data path salvaging”, in Proceedings of the Asia and South Pacific design automation conference (ASPDAC), 2011, pp. 437–442

[14] D. Fick, A. De Orio, J. Hu, V. Bertacco, D. Blaauw and D. Sylvester, “Vicis: a reliable network for unreliable silicon”, in Proceedings of the ACM/IEEE design automation conference (DAC), 2009, pp. 812–817.