Approaches to the SoC IP-Blocks' Design With Errors' Mitigation

Valentin Rozanov, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russia
valentin.rozanov@guap.ru, suvorova@aanet.ru

***Abstract*— Developing of failsafe systems is the problem solving in different sectors. Hardware design is not an exception. This sector imposes additional requirements constructing failover controllers, such as chip area and energy consumption. This article provides types and causes of errors to define their effect on SoC and offers the general idea of errors resilient SoC that consists of error detecting, error fixing and error mitigating mechanism. Also in the paper reconfiguration is considered as the error mitigation mechanism and methods used in practice to implement reconfiguration are presented. Before producing controller or other IP-block with error mitigation, first of all it is necessary to assess possible variants of failure and ways to improve its failsafe. Therefore methods of failure assessments are presented. As the result – Markov chain is chosen as the assessment tool and example of constructing of Markov chain for not reconfigurable and reconfigurable controller of transport layer protocol is presented. Comparative analyses of the results are carried out.**

I. INTRODUCTION

Failures can appear on different stages of controller's or other IP-block lifetime from developing till exploitation. On developing stages failures can be detected by verification and then removed. In the manufacturing process can appear failures also, that can be detected during test commissioning. These failures can be fixed (Fig. 1). However on the stage of using it is impossible to repair failure of memory or other blocks, or interconnections between blocks inside the IP-block performed on technology FPGA or ASIC.

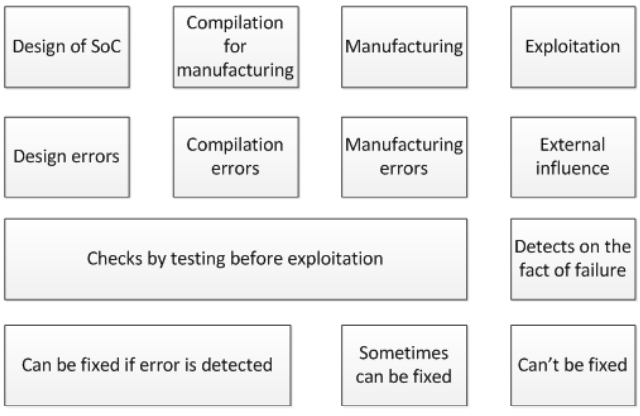

Fig.1. Errors on different stages of hardware controller or other IP-block lifetime

To understand the difference of errors in hardware controller or other IP-block part II consider types of errors and faults that can appear in the exploitation stage of controller lifetime. As the main aim is to make fault tolerant controller, part III describes construction of errors resilient SoC. In this paper was decided to explore problem of Hard-error mitigation. As a method of mitigation reconfiguration was chosen. So part IV considers practical variants of reconfiguration that are available in FPGA and ASIC technologies.

Before constructing reconfigurable system it's needed to assess how useful it's going to be. To understand which method is better a comparative analysis was done in part V. As the result of comparison Markov Chain was chosen to assess if reconfiguration is useful or not. For clarity the example transport layer protocol controller is used to make numerical evaluation of the reconfiguration results. Part VI describes the logic of controllers' work and the results of Markov Chain constructing for this protocol and the results of calculations.

II. TYPES AND CAUSES OF ERRORS

The main cause of faults and failures in SoCs for aerospace applications are space radiation, mesons and alpha particles. The main types of errors that are caused by these factors:

- Transient faults (Soft errors);
- Permanent faults (Hard errors).

Soft errors are:

- Single event upset (SEU) – changing a value stored in the flip-plop (SRAM cell) to the opposite value;
- Multiple cell upset (MCU) – changing the values of several neighboring memory cells;
- Single event transient (SET) – occurrence of a glitch at a transistor output;
- Single event functional interrupt (SEFI) – a soft error in a component that has an impact on functionality of the whole system; for example, an error that occurred in the processor program counter.

Single event upset (SEU) also known as Upset or bit-flip. In this case as a result of exposure to the active particles the value in the trigger (SRAM) changes in the .opposite. Error refers to a class of short-term, if in the future it is possible to write a trigger the correct value. If between the time when value of the

trigger was damaged and new value was written there was no reading operation from this trigger, error will have no affect the operation of the system. In some cases, the trigger is no longer possible to write another value - this error can be referred to a class of constants.

Multiple error (MCU) can occur in two cases:

- before undergoing SRAM cell in SEU, was overwritten correct value, one of the neighbor cells also subjected to SEU effects;
- One active particle had an impact on neighboring cells SRAM (the likelihood of this situation depends on the energy of the particle and the density of the memory cells of placement).

In both cases, there are two simultaneous system errors, i.e. correction coding single mistake will not be enough to restore the situation.

Single event transient (SET) occurs as the result of active particles impact on transistor. On the output of it forms a short voltage drop (glitch). So during correct value of voltage corresponding to the logical "0", for a short time high level voltage occurs that corresponds to the logical "1" (positive); during correct value of voltage corresponding to the logical "1", for a short time voltage level occurs that corresponds to the logical "0" (negative).

Hard errors are:

- Single event latch-up (SEL) – dramatically increases the leakage current;
- Single event gate rupture (SEGR) – the breakdown of the transistor's insulating layer by the active particle.

In the event of short-term errors, after some time the correct functioning can be restored. Constant errors occur as a result of irreparable damage.

Defined types of errors, consider the construction of a controller that can mitigate the consequences of these errors.

III. Construction of errors resilient SoC

The controller or other IP-block that is able to mitigate the consequences of errors should be provided with a system that detects and eliminate errors, which should be a part of the developed IP-block. The scheme of such system is shown in Fig. 2. In the state 1, the system is functioning properly. Status D- is parallel wist state 1 operates and monitors the output parameters of the system. If an error is detected at the output of the system, the error is fixing if it's possible, or a decision is making that it is impossible to correct the error. In the latter case - we believe that there is a hard error in the controller (SoC). One way to eliminate the hard error is constructing of reconfigurable systems.

State 2 is error fixing state – so it can be said that it's soft error state. State 3 – reconfiguring of the system – it can be said that it's hard error state. After each of these states system moves to state 1 and becomes able to work. However after error fixing (state 2) in state 1 returns initially working system, and after reconfiguration (state 3) in state 1 returns new system. So this changes need to be taken into account in the detection system

Let's define the concept of reconfiguration as a way to eliminate the consequences of Hard Error and options for its implementation.

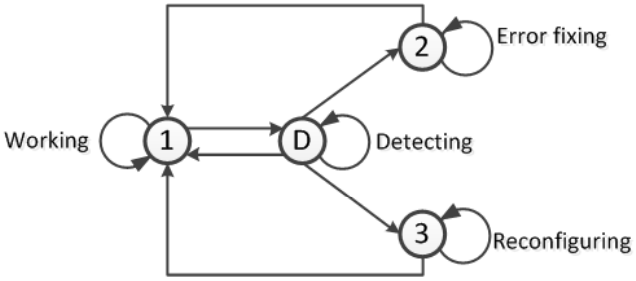

Fig. 2. Scheme of SoC with error detection and fault mitigation

IV. Reconfiguration as a fault mitigation methods

When hard error happens and when the IP-block is implemented NOT reconfigurable, failure of individual elements (memory or communication lines inside the IP-block), leads to the complete failure of the unit

If IP-block is implemented with reconfiguration ability, fault element can be turned off, or other connection lines can be used (if they are connected with fault block) to keep controllers or other IP-block functionality [1].

Dynamically reconfiguring SoC can be implemented with using of technologies of FPGA and ASIC.

When FPGA technology is used dynamical reconfiguration is achieved by partial replacement of the firmware. In many modern FPGA, for example, FPGA Xilinx Virtex possibility of dynamic reconfiguration is provided.

The area where CLB are situated is divided into frames – as usual vertical columns width of several CLBs (it depends of FPGA's type) and with length from top to the bottom border of CLB zone [1], [2], [3]. For each of these columns during system processing reload of firmware can be used individually. For reloading "partial" bitfile is used. It includes configuration only for selected one or several frames. Reloading of "partial firmware" can be made through dedicated configuration port ICAP.

In many cases there is no necessity to reconfigure all parts of SoC. When system is reconfigured, new element need to be placed on the area where uploaded components were situated. According to this, designing new controller, FPGA on the logical level is divided into areas (zones) for which is possible dynamic reconfiguration and for which dynamic reconfiguration is prohibited. Zones can be marked using different approaches. In a number of existing projects dynamic reconfiguration is used only for communication systems [4], [5], [6].

When ASIC technology is used dynamic configuration can be can be achieved by the following means:

- switching on and off different elements, in this case redundancy at the level of components and connections is used;

- using of look-up tables;
- using of logical elements libraries, that allows reconfiguration of logic (logical element can perform various functions depending on configuration for example NAND, NOR, NOT).

Regardless of the chosen manufacturing IP-block is necessary to determine whether the reconfiguration be practical technology to increase the reliability of the controller (or other IP-block) or not. To do this, use mathematical methods.

V. Methods of failure assessment

Controllers (or other IP-block) projecting process needs to assess advantages and disadvantages of creating reconfigurable NoC and assess probability of failure certain elements separately and as a whole the entire NoC.

First of all projecting the system we need to analyze possible errors or failures during exploitation. There are three main methods to analyze faults of system:

1) Logical block-diagram method.
2) Fault tree method.
3) Markov chain method.

Each of the methods starts with analyzing possible faults of the system and detecting failure rates or probability of failure for every element of considering system. Unfortunately getting this information may be very difficult because this information can appear secret. In this case calculations presented in this paper are made with imagined values of elements failure probability that helps to see the result of calculations in a short time.

Logical block-diagram (1) methods second step is combining (logical addition and multiplication) of separate components failure probability's. However this method can appear very complex if analyzing chain consists of several logical devices, sensors and execution units connected in a single logical sequence.

The second step of fault tree method (2) will be creating of tree diagram of faults. Fault tree analyze is a special technique that is used to analyze and identify conditions and factors that causes appearing of certain unlikely event.

Fault tree has one unlike event - accident or incident which is caused by a set of lower-level events - errors and failures. These causal chain called scenarios.

For communication between the events operations "AND" and "OR" are used in the tree nodes. Operation "AND" means that the higher-level event occurs while necessary events happens in the same time. Operation "OR" means that the any of necessary event can occur. Actually tree analysis consists of determining the causes and their combinations that leads to the header event.

Method 3 also needs to build the tree, but this tree includes all possible states of system and ways to transfer between them. A system of differential equations can be formed based on this diagram. The result of solving them is probability to stay in each state at chosen time or discrete time. So Markov chain method can be called most accurate of presented methods.

The second step of Markov chains method is to consider the process of the device, as a Markov process. A state graph is constructed, that shows states in which device can be during his operation (or group of devices when they are connected) and the transitions between them. The graph includes all possible states of the devices, which may arise due to failure of any of the components, including the full stop state, due to repair and diagnostics (if they are provided).The intensity of the transition (or transition probabilities) between states are set. Based on the graph a system of equations is constructed, probability of being in the states is seen as a complete group of events. If graph is considered in continuous time the intensity of transition is used, and the system of differential equations is solved. In the case of the discrete time consideration, the system of equations is solved with the transition probabilities. In continuous-time probability of being able to be in a state will be considered as a function of time, and the discrete-time, the probability of being in state will be probability to be in considered state on the current step [7].

Taking into account all the factors affecting the reliability, using Markov chains allows numerically evaluate the possible probability of considered variant of device configuration failure. But this method of analyzing the reliability of the system is difficult and time consuming from a mathematical point of view.

System level of the controller was chosen to consider. At this level, the controller is a set of memory blocks and logic blocks for data processing.

VI. Example of using Markov chain method for non-reconfigurable and reconfigurable controllers

Consider the example of using Markov chain to assess controller's probability to fail. Fig. 3 presents scheme of protocol for transferring and receiving data controller without acknowledgment. Data come in and out through the "Network Interface". When receiving data "Reception control unit", "Reception memory unit" and "Received data processing unit" forms "Receiving branch". These blocks are used to receive data and put it into memory unit to store for further processing and transmitting to the upper layers through "Data exchange interface". On the other side "Transmitting branch" is situated. It consists of "Sending data processing unit" block for preparing data from upper layers to transmit, "Sending memory unit" block for storing data waiting for transition and "Sending control unit" block for transferring stored data through the "Network Interface".

Failure of one element leads to a failure of the entire branch. For example "Sending memory unit" break down with probability p_{mt}=0.002 and "Reception memory unit" fails with probability p_{mr}=0.002. For reconfigurable controller "Arbitrator" block is used. If one of the "Memory unit" blocks fails "Arbitrator" starts using memory block of the other branch.

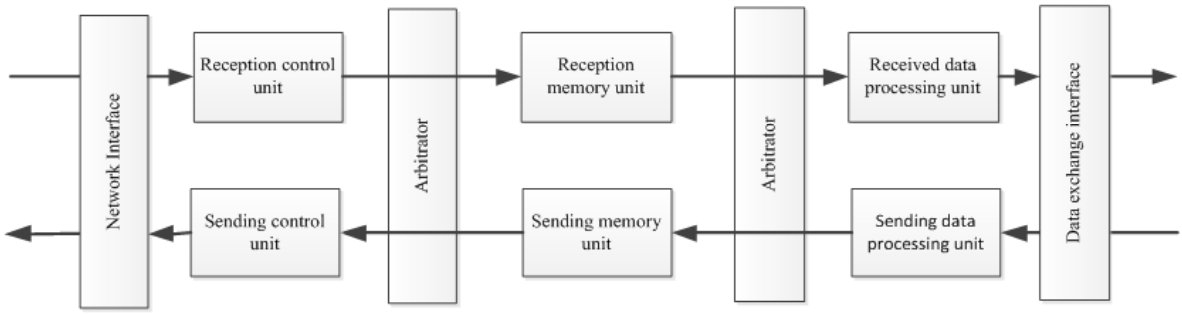

Fig. 3. Scheme of transport layer protocol controller

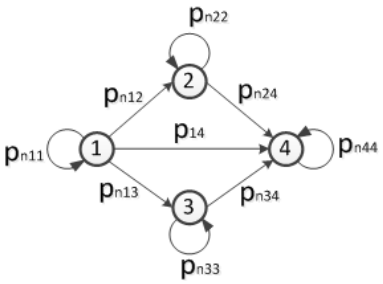

Fig. 4.Graph of controller states without reconfiguration

Fig. 4 presents graphs for non-reconfigurable (a) and reconfigurable (b) variants of controller. These graphs are Markov chains. Graph describes the following states:

1 – All works correct

2 – Receiving branch fails, transmitting branch works

3 - Transmitting branch fails, receiving branch works

4 for graph a и 6 for graph b – Both of branches fails

4 and 5 for graph b) – reconfigured state after states 2 and 3 (respectively)

The arcs of the graph are signed by the transition probabilities. Graphs are transient and absorbing [8]. Consider discrete time and using Chapman-Kolmogorov equation [9] the controller values of probability finding in each of the state were calculated for values of memory blocks failure p_{mr}=0.001, and p_{mt}=0.002.

$$P_n^*(t) = P_n^* \cdot P_n^t \quad (1)$$

Based on [7,8,10] to calculate possibilities of system to be in one of the states can be used equation (1)

Where:

P*(t) – probability of system to be in each state at the moment t, т.е. $P_n^*(t)=[P_n^*1, P_n*^2, P_n^*3,\ldots, P_n^*s]$, s>0 and is defined by the number of system's states. P make a complete group of independent events, in this way $P_n^*1 + P_n^*2 + P_n^*3 + P_n^*4=1$;

t – discrete time value;

P – matrix of transition probabilities from a state i to state j.

Probability matrix for the graph that describes the states of presented controller is a square matrix 4X4 (2)

$$P_n = \begin{bmatrix} p_{n11} & p_{n12} & p_{n13} & p_{n14} \\ 0 & p_{n22} & 0 & p_{n24} \\ 0 & 0 & p_{n33} & p_{n34} \\ 0 & 0 & 0 & p_{n44} \end{bmatrix} \quad (2)$$

Probabilities p_{n11}, p_{n22}, p_{n33} and p_{n44} shows the probability of system to be in one of the states on the current step (t).

At the moment t=0 probabilities are distributed as follows – $P_n^*(0)=[1,0,0,0]$, it means that system is in state 1 at the initial time.

To calculate the transition probabilities, addition and multiplication rules are used. Values of p_{mr} and p_{mt} were defined earlier. As the result following values of the transition probabilities from the state 1 received (3-5)

$$p_{n12} = p_{mr}(1 - p_{mt}) \quad (3)$$

$$p_{n13} = p_{mt}(1 - p_{mr}) \quad (4)$$

$$p_{n14} = p_{mt} \cdot p_{mr} \quad (5)$$

As transition probabilities forms a complete group of independent events the equation for p_{11} is (6)

$$p_{n11} = 1 - (p_{n12} + p_{n13} + p_{n14}) \quad (6)$$

Transfer probabilities for states 2 and 3 to get state 2 will be equal probabilities of faults of the last working memory block in the current state (7-10)

$$p_{n24} = p_{mt} \quad (7)$$

$$p_{n22} = 1 - p_{n24} \quad (8)$$

$$p_{n34} = p_{mr} \quad (9)$$

$$p_{n33} = 1 - p_{n34} \quad (10)$$

Considered graph is transient and absorbing because after finite number of steps states are left and getting new state system can't get to the previous one, and there is the state at which the process is terminated (state 4 – controller full fault). With absorbing state we assume that the probability of being in this state when hit is equal to 1, $p_{44}=1$ [10].

Let's calculate transfer probabilities assuming that $p_{mr}=0.001$, and $p_{mt}=0.002$. With the values of the transition probabilities, and value of the initial probabilities $P^*(0)$ becomes possible to define number of steps (the value of t), that is needed the system to get probability to be in states $P_n^*(t)=[P_n^*1<0.1, P_n^*2<0.1, P_n^*3<0.1, P_n^*4>0.99]$

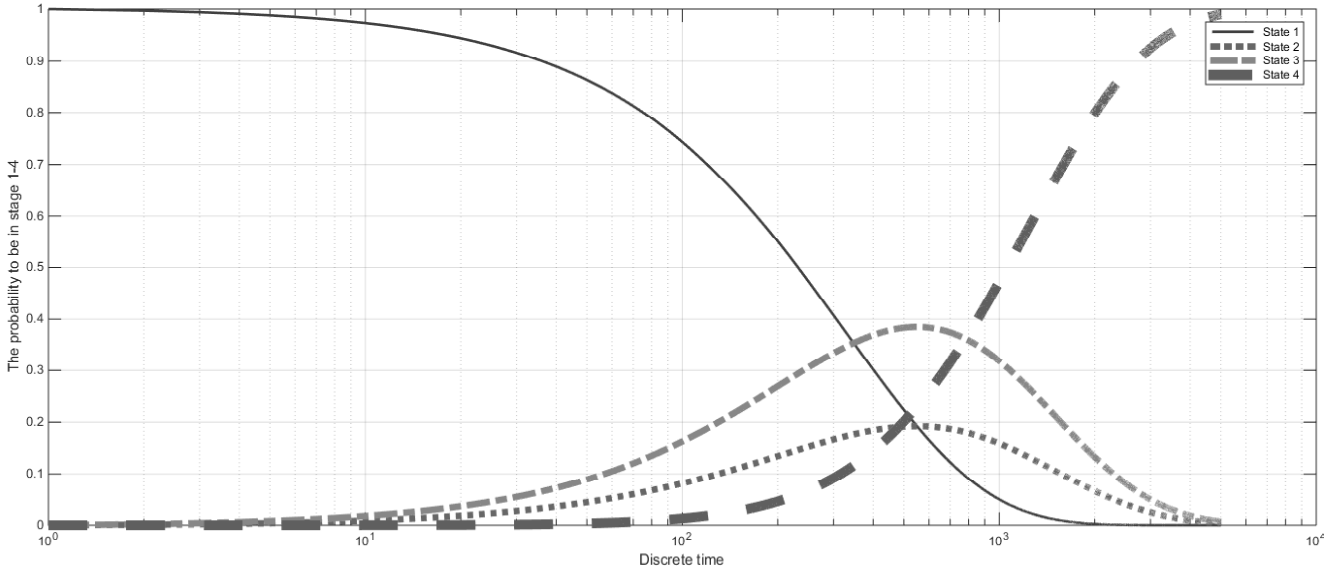

Fig. 5. Dependence of probability value to stay in state 1-4 from the step (t) value (non-reconfigurable controller)

Fig. 5 presents the graph of probability changing depending on the value of "t" (number of step) for non-reconfigurable controller.

As the result the number of steps to get probability equal 0.99 to stay in state 4 is $t_n=5009$.

Now let's explore controller with reconfiguration mechanism (Fig.6).

The controller operates similarly not reconfigurable. However, in case of one of the memory units faults system will reconfigure itself to work with one healthy memory unit. It will be used for both brunches. In this case systems the system is able to receive and transmit data even with one failed memory unit. To implement this scheme units "Arbitrator" are used. They controls and rule memory access for reading and writing from each of the brunches.

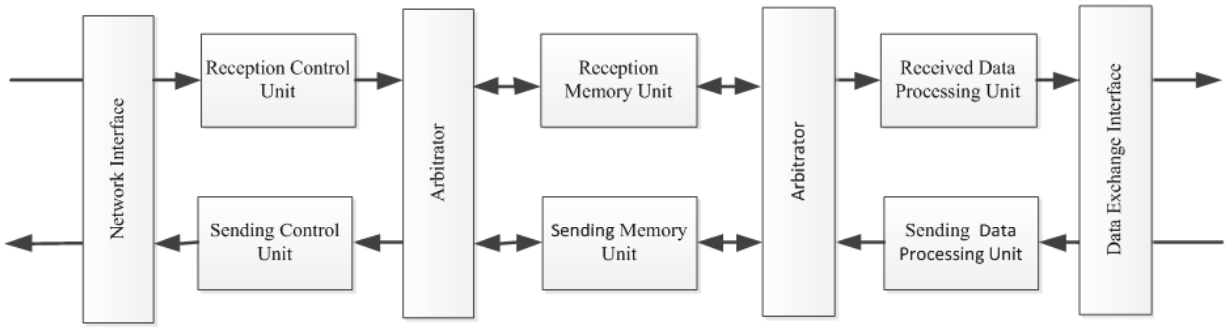

Fig. 6. Scheme of transport layer protocol controller

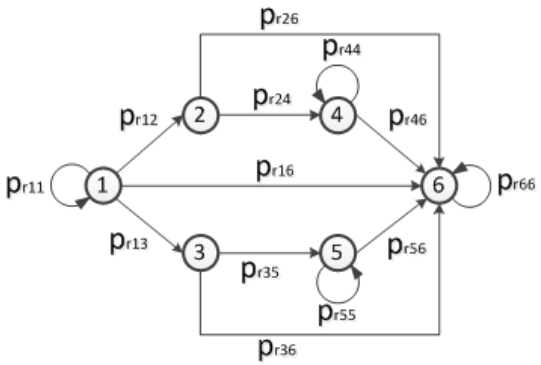

Fig. 7.Graph of controller states with reconfiguration in states 2 or 3

In the resulting state graph (fig. 7) sates 1,2,3,6 are the same as state 1,2,3,4 in state graph for not reconfigurable controller respectively.

State 4 is reconfiguration state for using transmitting memory unit for both brunches and state 5 – reconfiguration state for using receiving memory unit for both branches. The system is expected that if unit fails reconfiguration takes place immediately, so situation when system stays in state 4 or 5 for some time (for some number of steps for discrete time) is not considered. For the presented graph transition probabilities matrix will look (11)

$$P = \begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & 0 & 0 & p_{r16} \\ 0 & 0 & 0 & p_{r24} & 0 & p_{r26} \\ 0 & 0 & 0 & 0 & p_{r35} & p_{r36} \\ 0 & 0 & 0 & p_{r44} & 0 & p_{r46} \\ 0 & 0 & 0 & 0 & p_{r55} & p_{r56} \\ 0 & 0 & 0 & 0 & 0 & p_{r66} \end{bmatrix} \quad (11)$$

Transferring probabilities from state 1 can be defined as presented in equations 3-5 (12-15).

$$p_{r12} = p_{mr}(1 - p_{mt}) \quad (12)$$

$$p_{r13} = p_{mt}(1 - p_{mr}) \quad (13)$$

$$p_{r16} = p_{mt} \cdot p_{mr} \quad (14)$$

$$p_{r11} = 1 - (p_{r12} + p_{r13} + p_{r16}) \quad (15)$$

Because in case of failure is supposed to move in the reconfigured condition, and given that the graph is seen in a discrete form, the probability of stay for some time (some steps) in states 2 and 3 are not considered, and considered options for the transition from these states into the state 6 (total failure of all devices) (16, 17) or state 4 and 5 respectively (reconfigured condition).

$$p_{r26} = p_{r12} \cdot p_{mt} \quad (16)$$

$$p_{r36} = p_{r13} \cdot p_{mr} \quad (17)$$

As transition probabilities from 2 to 4 and 6 (as from 3 to 5 and 6) forms a complete group of independent events the equation for p_{r24} and p_{r35} are (18,19)

$$p_{r24} = 1 - p_{r26} \quad (18)$$

$$p_{r35} = 1 - p_{r36} \quad (19)$$

Similarly, the justification of the formulas 7-10 obtain the probability of transition from state 4 and 5 in state 6 and probabilities to stay in the current state on the next step (20-23)

$$p_{r46} = p_{mt} \quad (20)$$

$$p_{r44} = 1 - p_{r46} \quad (21)$$

$$p_{r56} = p_{mr} \quad (22)$$

$$p_{r55} = 1 - p_{r56} \quad (23)$$

State 6 is absorbing state, so probability to stay in this state is equal 1.

As in equation (1) let's use equation (23) to calculate probability to stay in states 1-6. Calculate values of this probabilities considering that (24), and until $P^*_{r\,6}$=0.9 with starting probabilities distribution (at moment t=0) $P^*_r(0)$=[1,0,0,0,0,0].

$$P^*_r(t) = P^*_r \cdot P^t_r \quad (24)$$

$$P^*_{r1} + P^*_{r2} + P^*_{r3} + P^*_{r4} + P^*_{r5} + P^*_{r6} = 1 \quad (25)$$

Values of probabilities p_{mr} и p_{mt} are the same, used for not reconfigurable controller. Fig. 8 presents the graph of probability $P^*_{r\,s}$ changing depending on the value of "t" (number of step) for reconfigurable controller.

Number of steps to get probability equal 0.99 to stay in state 6 is tr=4551 fo reconfigurable controller. Comparing results of calculating it's easy to see that non-reconfigurable model worked on 10% longer time (steps) than reconfigurable. However the first model worked partially (only one of branches was working). In the second situation controller worked with both branches, so it can be said that it was "completely serviceable".

VII. CONCLUSION

It should be noted that important role of receiving results of probability distribution plays form of the graph, values of the transition probabilities and starting conditions. Besides that in the presented model discrete time was used that can't take into account time of reconfiguration and time of fail detection which may vary according to complicity of controller and operating conditions.

The other notice that can be made is that if controller uses one memory block for two branches (to receive and transmit) he needs more time for data processing. In this case it's needed to understand how it will effect on the speed of data transferring, as we are talking about transport layer protocol controller.

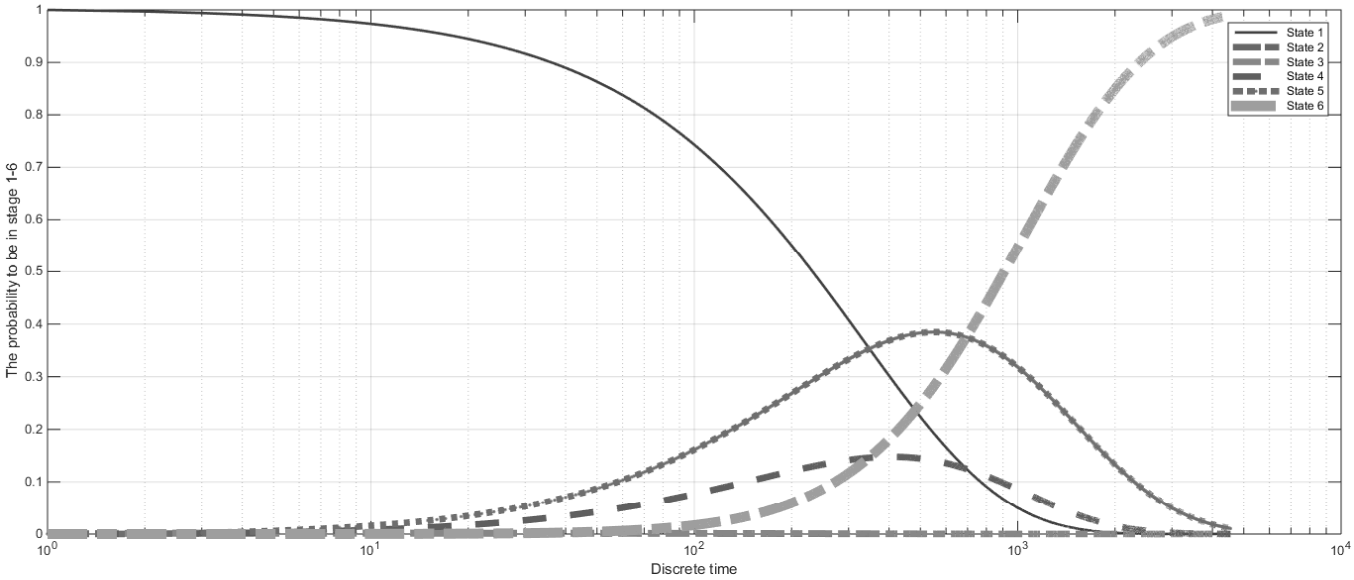

Fig. 8. Dependence of probability value to stay in state 1-6 from the step (t) value (reconfigurable controller)

Answering this question relatively to the presented controller it can be said, that if speed of data transferring will not be very high and controller will process data faster than transferring speed there will be no delays in the network. However if data transferring speed will be higher than data processing one, there will be a delay. The length of this delay depends on difference between transferring and processing speeds. But projecting controller it's better to know possible variants of delay before controller's implementing. Considering of this problem is planned as future work.

Talking about disadvantages of presented variant of protocol and its reconfiguration, it could be said:

- After reconfiguration speed of data receiving and transmitting will be lower, because of using one memory unit for two directions;
- If the last memory unit breaks down, controller becomes faulty in a moment.

Advantages of this variant are:

- Ensure full operability of the controller even in the event of failure of one of the memories that will probably be more important than the preservation of the processing speed;
- Maintaining the required space occupied by NoC in terms of memory elements.

Presented variant of protocol is used as an example only and explains variant of reconfiguration. Each variant of reconfigurable system needs to be considered accordance with the purpose of use, chip area, power consumption requirements.

Summing up further conclusions could be made:

- Configuration of Markov chain depends on the systems number of states. These states rely on configuration specific controller implementation;
- When configurable controller is considered chain becomes more complicated
- The best way to consider reconfigurable systems is to construct and count Markov chain with continuous time.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation according to the base part of the state funding assignment in 2016, project № 1810.

REFERENCES

[1] R.N.Pittman, *Partial Reconfiguration: A Simple Tutorial Technical Report*. 2012. pp. 158 – 235.

[2] W.Lie, W.Fengyan, *Dynamic partial reconfiguration in FPGAs, Third International Symposium on Intelligent Information Technology Application*. 2009. pp. 445 - 448.

[3] Partial Reconfiguration Design with Plan Ahead, Web: http://www.xilinx.com.

[4] V.Rana, D.AtienzaMarco, D.Santambrogio, *A Reconfigurable Network-on-Chip Architecture for Optimal Multi-Processor SoC Communication*, IFIP AICT 313. 2010. p. 232–250.

[5] E.Suvorova, Y.Sheynin, N.Matveeva, Reconfigurable NoC Development with Fault Mitigation, *FRUCT 18*, 2016

[6] E.Suvorova,, Y.Sheynin, N.Matveeva. Fault Mitigation in reconfigurable NoC routers with thin design rules, International Journal of Embedded and Real-Time Communication Systems, *Volume 6, Issue 1, January-March 2015*, Pages 28-46

[7] M.Ya.Kelbert and Ju.M.Sukhova Probability and statistics in examples and problems. Vol. II Markov chain as the starting point of the theory of random processes and their applications Moscow, 2009

[8] U.N.Fedorov *Engineer Manual: Design and development. Educational and practical guide*. Infra-Engineering. 2008.

[9] V.I.Tihonov and M.A.Mironov, *Markov process*, 1977, p. 238

[10] Ricky W. Butler and Sally C. Johnson, *Techniques for Modeling the Reliability of Fault-Tolerant Systems With the Markov State-Space Approach*, NASA Reference Publication, Langley Research Center • Hampton, Virginia

[11] T.E.Aliev *Basics of discrete systems modeling*, Saint-Petersburg, 2009. p.168.