Multi-version Digital Systems Evolved with Genetic Algorithms: Designing and Estimating Fault Tolerance

Vyacheslav Kharchenko, Nataliya Yakymets

Abstract – **The paper discusses the main challenges in multi-version system design. Most of the current techniques for design diversity rely on a system life cycle within a single design concept, so it is not possible to make a system robust to the defects that are common to each version since widely used Computer Aided Design (CAD) tools usually have only standard algorithms and conventional logic implemented. In this work, we give a solution to the problem of achieving the least correlation level within the multi-version projects by using different approaches to system design. Focusing on the non-classical design based on the genetic algorithms (GAs), we represent the metrics to assess a diversity level in multi-version digital systems and suggest several methods to estimate the reliability of such systems. We prove those by an example of heating controller for AN-70 plane as well.**

Index Terms – **Fault Tolerance, Multi-Version Systems, Design Diversity, Genetic Algorithms, Digital Automata.**

I. Introduction

Field experience with so-called critical systems such as airborn control systems or atomic power plants safety systems shows that they are quite sensitive to the faults caused by design and physical defects [1-3]. This problem can be tagged as an increasingly challenging issue for the industry as a whole, mostly because of the tendency to develop critical applications as complex System on Chip (SoC) designs. To reduce the overall number of defects, the multi-version approach to system design can be successfully used. It means developing several versions of the same system which must be as different as possible [4-8]. For this reason, several sufficient methods for SoC-oriented architectures use standard Computer Aided Design (CAD) tools to diversify the design [5, 6]. Nevertheless, despite the effectiveness of these methods most of those manage the life cycle diversity within a single design concept, so it is not possible to make a system robust to the defects that are common to each version since widely used CAD tools usually have only standard algorithms and conventional logic implemented. Code inaccessibility and unavailability make it impossible to predict a behavior of CAD tool and prevent an occurrence of the defects. To avoid such a difficulty, the application of non-classical design is suggested in order to develop non-conventional digital systems [9, 10, 11, 12]. It exploits the ideas from nature and human way of thinking instead of the apparatus of integral and differential calculus. Nowadays, evolvable hardware applications are maturing and seeing deployment into the real-world as fielded applications. In [9, 10], development of digital systems as an artificial neural network has been described, some other works discuss the utilization of the genetic algorithms (GAs) in digital [12, 13] and analog [14] system design, robotics [15] and even in the domestic life [16]. In [17], it has been pointed out that one of the most effective ways to develop a multi-version fault tolerant system is to combine classical design based on using CAD tools with non-classical ones in order to significantly increase a possibility of receiving the least correlated versions. Nevertheless, the actual problem is how to elaborate the strategy which allows obtaining the highest diversity level in a multi-version project.

In this paper, we focus on GAs as an alternative to the classical design because they are targeted for finding exact or approximate solution to problems connected with digital system design. Often GA gives only approximate solution by producing versions (individuals), which have correct output states only for a subset of the input data set. In this case, the common practice is to gather these partially correct versions in the majority architectures [18]. One of the serious disadvantages of such an approach is that it does not take into account known information about the correctness of each individual. So, very often a redundant number of partially correct individuals (or partially correct automata, PCA) is involved resulting in growing the system complexity. In our previous works [19], we discussed a dependency between the complexity of system logic and time required to develop a digital system with GA. We suggested "Sliding Testing" technique to reduce time consumptions that meant testing only a part of input and output data while calculating the fitness of individuals in a population. In this case the evolved versions will be partially definite (partially definite automata, PDA), so they have to be gathered in the redundant architectures to provide the full definiteness of a system. At the same time, as we know information about the definiteness and correctness of every version beforehand, we can manage a reliability level for the systems implemented with PDA and PCA by adding new automata with definite and correct output data that corresponds to the certain input data.

Vyacheslav Kharchenko is with the National Aerospace University named after N.E. Zhukovsky "KhAI", Kharkiv, Ukraine (e-mail: V.Kharchenko@khai.edu)
Nataliya Yakymets is with the National Aerospace University named after N.E. Zhukovsky "KhAI", Kharkiv, Ukraine (e-mail: nataliya.yakymets@informatik.uni-stuttgart.de)

In this paper, we develop a strategy of project diversification for digital systems with SoC architecture and suggest several methods to estimate reliability of such systems evolved with GA.

The remainder of the paper is organized as follows. Section 2 elaborates the strategy to achieve the least correlation level within a multi-version project. In section 3, we continue with the case study describing the development of fault tolerant systems with PDA and PCA. We estimate the reliability of such systems in section 4. Finally, we prove our methods in section 5 by the example of heating controller for AN-70 plane.

II. External and Internal Design Diversity

The application of multi-version approach to the system design assumes obtaining version redundancy by varying the set of resources used in design process to receive the most alternate versions of the same project. For example, several CAD packages or several developer groups can be involved into the design flow. In order to get an n-version project, it is necessary to isolate n subsets of resources that allow implementing the same functionality. The versions obtained from the different subsets will be less correlated than those that are received by varying characteristics within the only one subset. For example, the lesser correlation level can be achieved with several CAD tools rather than with different characteristics of a single CAD package. So, the design diversity can be observed from the several levels of system design (Fig. 1). So-called internal design diversity assumes obtaining alternative versions from only one isolated subset of resources used in design. The external one means the application of several sets of resources.

In fact, the major challenges in multi-version system design are:

1) isolation of the subsets with the maximum cardinality;

2) selection of the diversity metrics, which allow comparing system versions;

3) risk analysis while estimating the compatibility of versions received with the different subsets of resources.

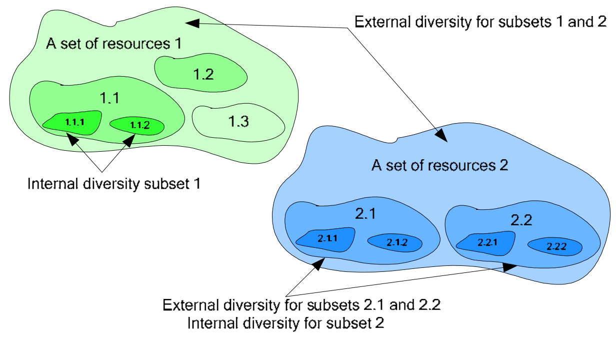
A set of resources 1
1.1
1.2
1.3
1.1.1
1.1.2
Internal diversity subset 1
External diversity for subsets 1 and 2
A set of resources 2
2.1
2.2
2.1.1
2.1.2
2.2.1
2.2.2
External diversity for subsets 2.1 and 2.2
Internal diversity for subset 2

Fig. 1. External and internal design diversity

At this point, it seems that one of the effective ways to obtain the most diversified project is to exploit the external design diversity based on the different approaches to system design and use several non-classical approaches such as GAs or neural networks along with the classical one.

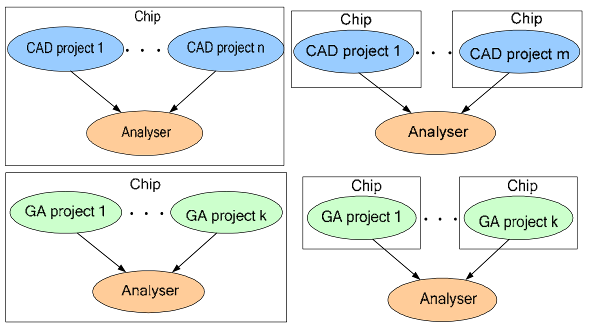

Fig. 2. Internal diversity that uses different design approaches

In this work, we are concerning GA as an alternative to the classical system design, mainly because of the possibility to get non-conventional solutions in system design and going towards the reliability of digital systems developed with GA. Thus, if standard CAD tools are used the different versions can be obtained at the following phases: hardware selection, project entrance, compilation, testing and verification. In case of the GA application, the project diversity can be achieved at the phases of GA presetting, selection, crossover, mutation and inversion of individuals. Combination of CAD-based and GA-based approaches allows considering cases where both internal and external design diversities are possible (Fig. 2-3). Moreover, version implementation for a single chip or for a number of chips gives an additional opportunity to utilize spatial diversity in system design.

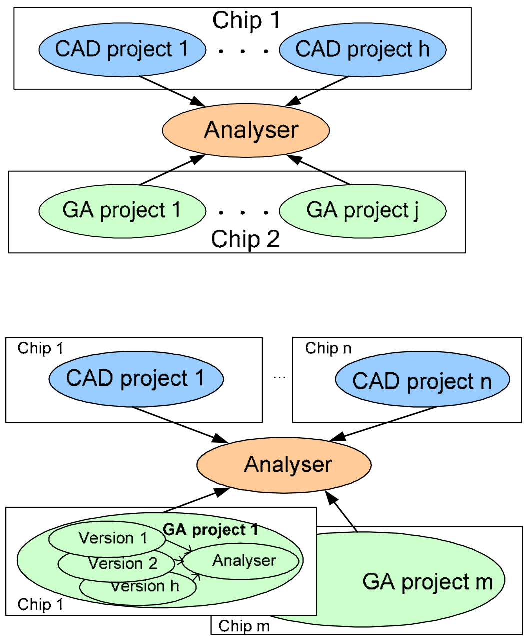

Fig. 3. External diversity that uses different design approaches

Versions evolved with GA can be even used to control the functionality of a multi-version system developed with the standard tools (Fig. 4).

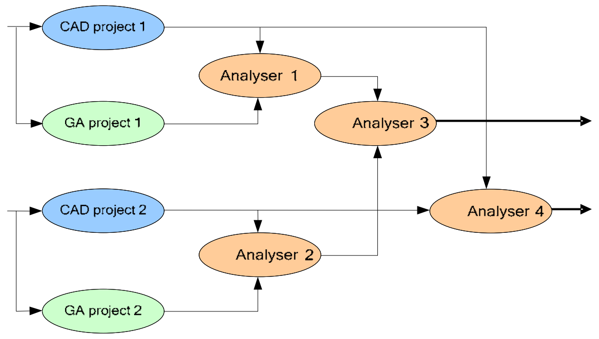

Fig. 4. Using versions obtained with GA as the control ones in a multi-version project

To assess a diversity of such multi-version projects we suggest comparing them at two levels. At the *logical level*, a digital system with SoC architecture can be represented as a graph G={L, N}, which is constituted by N logic cells and a set of cell interconnections, L. Therefore, the degree of distinction for graph nodes allocation (set of nodes N_1 and N_2) can be used as a diversity metrics (Fig. 5):

$$M_1 = \frac{|N_1 \cap N_2|}{\frac{|N_1|+|N_2|}{2}}, \qquad (1)$$

where |N1|, |N2|, |N1 ∩ N2| are the capacities of correspondent sets.

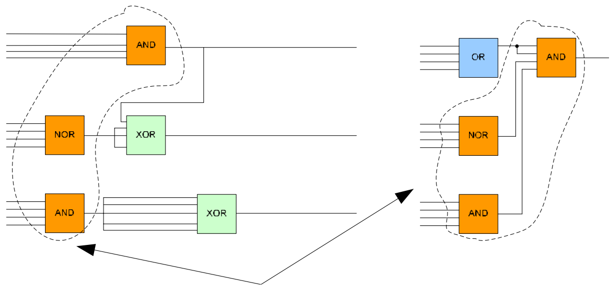

Figure 5. Graphs that show cell localization in a chip

At the *physical level*, a digital system with SoC architecture can be represented as a spatial configuration, which is constituted by N logic cells on a chip. A set of logic cells corresponds to set $S = \{s_i\}_{i=1}^{N}$, where s_i are the coordinates of the i-th cell. At this level, a degree of distinction for logic cell topology in a chip can be selected as a diversity metrics (Fig. 6):

$$M_2 = \frac{|S_1 \cap S_2|}{\frac{|S_1|+|S_2|}{2}} \qquad (2)$$

where |S1|, |S2|, |S1 ∩ S2| - are the capacities of correspondent sets.

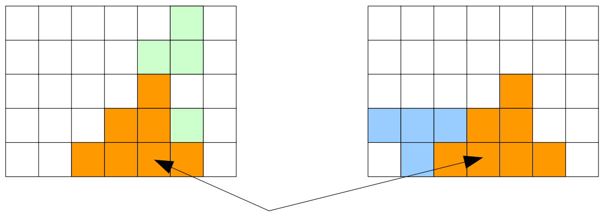

Fig. 6. Topology of logic cells in a chip

III. Partially Definite and Partially Correct Automata

In this section, we classify versions (automata) that can be evolved with GAs, and investigate the ways to develop fully correct and definite system.

The digital automaton can be described with its level of definiteness and correctness. The correctness shows how the functionality of automaton fits to its specifications. The definiteness means a level of preliminary awareness about the correctness of automaton. Thus, digital automata can be fully or partially correct and fully or partially definite [19].

Partially correct automaton (PCA) is an automaton where at least one input state x_i in the set of input data X does not correspond to required output state y_i in the set of correct output data Y_c, i.e. $\exists x_i \in X: x_i \rightarrow y_i, y_i \notin Y_c \subset Y$, where x_i – current input state; X – set of input data; y_i –current output state that corresponds to x_i; Y_c – set of correct output data; Y – total set of output data.

Partially definite automaton (PDA) is an automaton where at least one input state x_i in the set of input data X corresponds to such an output state y_i where information about its correctness (whether y_i is in Y_c or not) is not known beforehand, i.e. $\exists x_i \in X: x_i \rightarrow y_i, y_i \notin Y_s, Y_s \subset Y$, where Y_s – set of definite (specified) output data.

In fact, the fully correct system can be composed of the several PCA or/and PDA in such a way that the total set of correct and definite output states of automata covers the total set of input data.

IV. Reliability of Digital Systems Implemented with Partially Definite and Partially Correct Automata

The architecture of digital systems implemented with the PDA and PCA allows estimation of their reliability using the apparatus of structural reliability theory.

The analysis of reliability for such systems includes the following steps:

1) dividing a set of input and output data of automata into groups as shown in Fig.7;

2) estimating the influence of automata failures on entire system;

3) selecting a primitive object to be used in the reliability analysis: the primitive objects can be automata, groups of definite and correct output data, automata and groups of definite and correct output data;

4) developing a reliability block diagram (RBD);

5) forming the equations to estimate system reliability with regard to its RBD.

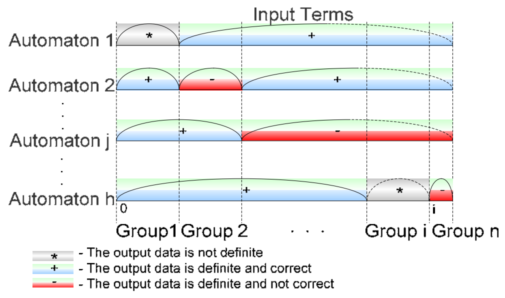
Input Terms
Automaton 1
Automaton 2
Automaton j
Automaton h
*
+
-
0
i
Group1
Group 2
Group i
Group n
* - The output data is not definite
+ - The output data is definite and correct
- - The output data is definite and not correct

Fig. 7. Allocating the groups within the set of input and output data if a system consists of the several PDA and/or PCA

If a digital system consists of one fully definite and fully correct version, its reliability is the same as the reliability of a non-redundant non-recoverable system. If a digital system is composed of several PDA or/and PCA it can be considered as a non-recoverable system with passive redundancy. To estimate the reliability of such systems, the additional analysis of their functionality must be performed to prove the choice of the primitive object and develop RBD more precisely.

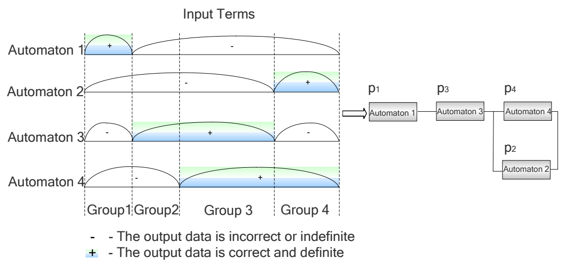

Fig. 8. RBD for a system consisted of four automata if primitive objects are automata

If the primitive objects of RBD are automata, we assume that the failure of every automaton results in an overall loss of its functionality. Such an assumption is reasonable for a few automata as the process of RBD development is quite complicated and non-conventional for each particular system. So, RBD is formed according to already known information about the automata behavior. The example in Fig. 8 shows that the system fails if Automaton 1 fails or Automaton 3 fails or Automaton 2 and Automaton 4 fail. The probability of no-failure for such a system is calculated with the following formula: $P(t)=p_1(t)\times p_3(t)\times(1-(1-p_4(t))\times(1-p_2(t)))\times p_K(t)$, where p_i is a probability of no-failure for automaton i; $p_K(t)$ is a probability of no-failure for the subsystem that implements switching between automata.

If the primitive objects of RBD are groups of definite and correct output data, the assumption is that the failure of an automaton does not result in the overall loss of its functionality and it can produce correct output data for some input terms. In contrast to the previous assumption, this one is reasonable in case a significant number of automata constitute a whole system. Nevertheless, there is one serious disadvantage: during the synthesizing with GA, a digital system is considered as a “black box” with certain inputs and outputs without any information about its internal structure. Therefore, very often information about the nature and consequences of the automaton’s failure is not available. The probability of no-failure for such a system is calculated with the following formula:

$$P(t)=\prod_{i=1}^{n}[1-\prod_{j=0}^{m}[1-p_{ij}(t)]]\times p_K(t)\,, \qquad (3)$$

where $p_{ij}(t)$ is a probability of no-failure for automaton j within the group of correct and definite output data i; n is a number of groups of correct and definite output data; m is a number of automata that have correct and definite output data within the group of correct and definite output data i.

For the example in Fig. 9, $P(t) = p_{11}(t)\times p_{23}(t)\times(1-(1-p_{33}(t)) \times (1- p_{34}(t))) \times (1-(1- p_{42}(t)) \times (1- p_{44}(t))) \times p_K(t)$.

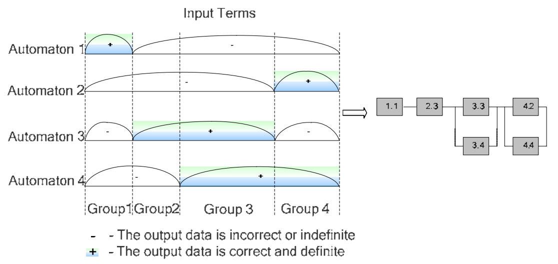

Fig. 9. RBD for a system consisted of four automata if primitive objects are groups of output data

By assuming the primitive objects of RBD to be automata and groups of definite and correct output data, we mean that the failure of an automaton might result in either the overall or partial loss of its functionality with the certain probability. The probability of no-failure for such a system is calculated using the following formula:

$$P(t)=P'(t)\times P''(t)\times p_K(t)= \\ =P'(t)\times(\prod_{i=1}^{n}(1-\prod_{j=0}^{m}(1-q_{ij}(t))))\times p_K(t), \qquad .(4)$$

where P'(t) is a probability of no-failure for those parts of automata which incorrect functioning results in an overall functionality loss of the appropriate automaton; P'(t) is calculated as if the primitive objects are automata, but in equation (6) $p'_j(t)$ is used instead of $p_j(t)$ that is the probability of no-failure for those part of automaton j those incorrect operating results in an overall functionality loss of this automaton; P"(t) is the probability of no-failure for those parts of automata which incorrect work results in a functionality loss of the appropriate automaton in the appropriate groups; $q_{ij}(t)$ is the probability of no-failure for automaton j within group i in case of no-failure operation of its part which fault results in the overall functionality loss of automaton j.

The example in Fig. 10 shows how the probability of no-failure is calculated for such systems: $P(t) = p'_1(t) \times p'_3(t) \times$ $\times (1-(1-p'_4(t))\times(1-p'_2(t))) \times q_{11}(t) \times q_{23}(t) \times (1 - (1 - q_{33}(t)) \times$ $\times (1- q_{34}(t))) \times (1 - (1 - q_{42}(t)) \times (1 - q_{44}(t)))\times p_K(t)$.

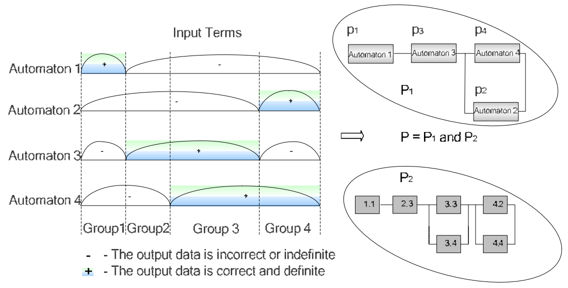

Fig. 10. RBD for a system consisted of four automata if primitive objects are both automata and groups of output data

V. Experimental Results

We have verified our approach using the example of the digital heating controller developed by means of the standard CAD tool, and currently used in AN-70 plane. We have implemented this project using C++, AHDL languages and Quartus II tool on a Pentium IV with 1500 Mhz clock and 1 GB RAM.

We have used a simple GA with a population size of 50, GA cycles of 1000, crossover probability of 0.75 and mutation probability of 0.25. We have also determined the modeling area as 4×4 array of logic cells of programmable logic device (PLD). We have chosen versions by estimating their diversity at the logical level according to (1).

Input data for the heating controller: 1-st bit determines a sign, 2-7 bits determine a value of the temperature (°C). Output data for the heating controller: '01' – the temperature is lower than 15°C, '10' – the temperature from 15°C up to 35 °C, '11' – the temperature is higher than 35°C.

Designing fault tolerant digital systems with the PDA and PCA includes two phases. The initial data for the *first phase* is a truth table of the required system as well as a type of automata that should be proved depending on the system complexity, cost and time constraints. During the design flow, several partially definite and partially correct versions of the heating controller, which are sufficient to form a fully definite and correct model, have been evolved with GA. Each version represents a digital system at the gate level coding a single variant of mapping and providing information about the interconnections between logic cells in PLD and their internal functions (Fig. 5). The received model of heating controller involves 9 logic cells of modeling area and includes two partially correct automata evolved in 405 and 789 populations (Fig. 11).

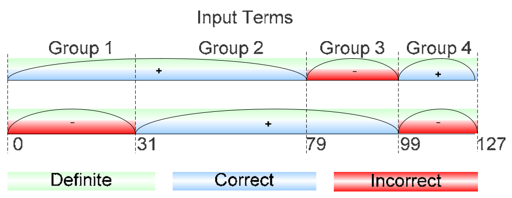

Fig. 11. The scheme of term overlapping in the heating controller model implemented with partially correct automata

We assume that each cell can realize one of the following internal functions: AND, OR, XOR, and it has one output and four inputs connected either to one of a global inputs (input variables from the truth table given for a system) or to an output of any cell. The fitness *fit* has been calculated for each version (individual) by comparing its output data to the existing project specifications (truth table). It equals a ratio between the number of correct terms of the individual and the overall number of terms from the truth table:

$$\mathrm{fit}_i = \frac{X_r \times 100\%}{2^n}, \qquad (5)$$

where *i* is a number of individual in a population; X_r is the total number of input terms that corresponds to the correct output data; *n* is a number of inputs.

In the *second phase*, a special block is developed in order to implement switching between automata and achieve full correctness of a system. The initial data for this phase is a number of automata utilized in the model and information about their behavior.

We have implemented the model in Fig. 11 to FPGA EP1K10TC144-3 (family ACEX 1K) using the CAD tool Quartus II v.6.0 and have compared it with the existing prototype. The scheme that allows switching automata has been designed according to Table 1 and Fig. 11.

To translate the model into the acceptable format in order to exploit the standard CAD tool, each automaton has been described in AHDL language as follows. The codes of the internal functions of each cell have been extracted from binary string that describes the behavior of automata as well as codes that show cell interconnections. Then we have defined the appropriate variable for each cell to implement its internal function and set the connections between these variables, inputs and outputs of automaton. We have received a hierarchical project by applying such a technique to every automaton included in the model: on the top level, there are a switching block and automata represented as "include files".

Table 1
Switching logic for two automata

	Information about current behavior of automata		
Known information about definiteness and correctness of automata	A - Automaton	A1 = A2	A1 ≠ A2
	A1 and A2 have definite and correct output	«OK», Switch to A1 or A2	«Fail»
	A1 has definite and correct output	«Risk», Switch to A1	«OK», Switch to A1
	A2 has definite and correct output	«Risk», Switch to A2	«OK», Switch to A2

To estimate the reliability of the developed project, we have chosen two parameters:

1) the probability of no-failure;
2) the probability of keeping system operating state.

The first parameter has been calculated for chip failure rate $\lambda=10^{-7}$ 1/hour, $p_K = 1$, time intervals t={10^1, 10^2, 10^3, 10^4, 10^5} hours and overall number of logic cells N=256. Time till the PLD fault has been given by the exponential distribution.

If the primitive objects are automata, the probability of no-failure is calculated with the following formula: $P(t)=p_1(t)\times p_2(t)\times p_K(t)$. We have assumed that $p_1(t)=p_2(t)$, so $P(t)=p(t)^2\times p_K(t)$. The probability of no-failure for a single automaton is equal to $\lambda_a= (\lambda\times N_a)/N$, where N_a is a number of logic cells that constitute single automaton. $N_a=N_s/h$, where N_s is a number of cells allocated for the whole system and h is a number of automata. Therefore, the probability of no-failure for the whole system is the following:
$$P(t)= e^{-2\times\frac{10^{-7}\times[9/2]}{256}\times t}.$$

If the primitive objects are groups of definite and correct output data, the probability of no-failure is calculated as $P(t)=p_{11}(t)\times(1-(1- p_{12}(t) \times(1- p_{22}(t)) \times p_{23} (t) \times p_{14}(t) \times p_K(t)$. We have assumed that all probabilities p(t) are equal, so $P(t)=p(t)^3\times(1-(1- p(t)^2)\times p_K(t)$. The probability of no-failure for a single group of correct and definite data of automaton equals $p(t)=e^{-\lambda_g t}$, where λ_g is a failure rate of automaton for a group g. λ_g is calculated as $\lambda_g= (\lambda\times N_g)/N$, where $N_g =N_s/(h\times n)$ and n is a number of groups. Therefore, the probability of no-failure for the whole system is the following: $$P(t)= e^{-3\times\frac{10^{-7}\times\left[\frac{9}{2\times4}\right]}{256}\times t}\times(1-(1-e^{-\frac{10^{-7}\times\left[\frac{9}{2\times4}\right]}{256}\times t})^2).$$

To compare the developed project of heating controller with the prototype, we assume prototype to be a single fully definite fully correct automaton. Thus, $P(t)=e^{-\lambda_p t}$, where λ_p is a failure rate of the prototype. λ_p is equal $\lambda_g= (\lambda\times N_s)/N$, so the probability of no-failure for the prototype is $$P(t) = e^{-\frac{10^{-7}\times 74}{256}t}.$$

The values of P(t) for the obtained project and its prototype are given in Table 2. The gain in reducing the probability of no-failure for both projects is shown in Table 3.

To analyse the reliability for the obtained heating controller, we have also used a PLD fault simulator [21] and assessed the probability of keeping system operating state for the several fault configurations. We have received the values of the probability of keeping system operating state by the serial experiments where different configurations of faults were generated. The experimental results given in Table 4 show that the system is able to keep its operating state even though there is a significant number of failed cells in the chip. The reason is the compactness of the developed project: it uses only 27 logic cells (Fig. 12), whereas in the heating controller that is currently used in AN-70 plane, the overall number of cells involved is 74.

TABLE 2
THE PROBABILITY OF NO-FAILURE FOR HEATING CONTROLLER AND ITS PROTOTYPE

	Primitive Object	Time, hour				
		10	10^2	10^3	10^4	10^5
New Project	Automata	0,999999965	0,999999648	0,999996484	0,999964844	0,999648499
	Groups of output data	0,99999999986816	0,99999999868164	0,99999998681641	0,9999998681647	0,99999868170829
Prototype		0,99999997109938	0,999999710 9379	0,999997109 4168	0,99997109 7927	0,99971135 48834

TABLE 3
THE GAIN IN REDUCING THE PROBABILITY OF NO-FAILURE FOR HEATING CONTROLLER COMPARING TO ITS PROTOTYPE

Primitive Object	Time, hour				
-	10	10^2	10^3	10^4	10^5
Automata	8,25892738	8,21199097	8,22122646	8,22109243	8,21178650
Groups	21,9252471	21,9258838	21,9256153	21,9228633	21,8953904

TABLE 4
THE PROBABILITY OF KEEPING SYSTEM OPERATING STATE WITH THE SEVERAL NUMBERS OF FAULTY CELLS

Faulty Cells	1 Cell	2 Cells	3 Cells	5% of Chip	10% of Chip	25% of Chip	50% of Chip	75% of Chip
GA-project	0.958	0.951	0.943	0.865	0.810	0.663	0.398	0.080
Prototype	0,870	0,797	0,745	0,480	0,343	0,050	0	0

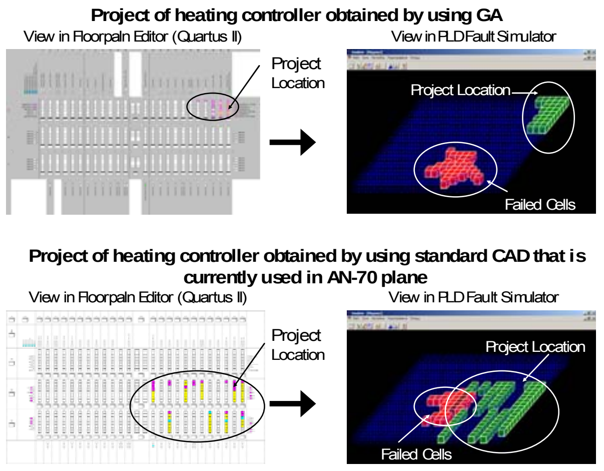
Project of heating controller obtained by using GA
View in Floorpaln Editor (Quartus II)
View in PLD Fault Simulator
Project Location
Project Location
Failed Cells
Project of heating controller obtained by using standard CAD that is currently used in AN-70 plane
View in Floorpaln Editor (Quartus II)
View in PLD Fault Simulator
Project Location
Project Location
Failed Cells

Fig. 12. Physical location of the heating controller projects in the FPGA EP1K10TC144-3 and their representation in the PLD fault simulator

In fact, both prototype and project evolved with GA can be gathered to duplex architecture as it is shown in Fig. 3 to constitute a multi-version project of the heating controller. Another way is to use the obtained project as a control module for the prototype (see Fig. 4).

VI. Conclusion

In this work, we have elaborated the overall strategy that allows a developer to manage the level of design diversity while developing multi-version projects of digital systems with SoC architecture. We have found out that one of the most efficient ways to obtain a high level of system diversity is to use different approaches to the system design. Focusing on the design with GA, we have suggested several methods to estimate reliability of systems evolved with GA that consider information about the definiteness and correctness of each system version. We have illustrated them by estimating the reliability of heating controller evolved with GA and by comparing it to its prototype. Further research can focus on the issues of version selection to minimize probability of common mode failure and its assessment taking into account implementation of design into a chip. Besides, we are planning to develop tools for support of design decision making.

References

[1] Blanke, M., Kinnaert, M., Lunze, J., Starosweicki, M. Diagnosis and Fault-tolerant Control. Springer, 2006, pp. 672.

[2] Stevens, L., Lewis, F. Aircraft Control and Simulation. Wiley-IEEE, Technology & Industrial Arts, 2003, pp. 680.

[3] Pasetti, A. Software Frameworks and Embedded Control Systems. Springer, Computers, Languages Programming, 2002, p.293.

[4] Townend, P., Jie Xu, Munro, M. Building Dependable Software for Critical Applications: Multi-Version Software versus One Good Version. In Proceedings of the 6th Int. Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'01), 2001, p. 103.

[5] Avizienis A., Lapric J.C. Dependable Computing: From Concepts to Design Diversity // Proceedings of the IEEE, 1986, vol. 74, issue 5, pp. 629- 638.

[6] Kharchenko, V., Tarasenko, V., Ushakov, A., The Fault-tolerant PLD-based Embedded Digital Systems, National Airspace University "KhAI", Kharkiv, 2004, pp.207.

[7] Hatton, L. N-version Design Versus One Good Version. In Proceedings of Software, IEEE, 1997, pp. 71-76.

[8] Anderson, S., Felici, M. Safety, Reliability And Security Of Industrial Computer Systems. Reliability Engineering & System Safety. 2005, vol. 89, issue 1, pp. 1-5.

[9] Shuqing Wang, Jiaping Liao, Zipeng Zhang, Xiaohui Yuan. Application of Neural Networks and Genetic Algorithm in Knowledge Acquisition of Fuzzy Control System. In Proceedings of the 6th World Congress on Intelligent Control and Automation. 2006, vol. 1, pp. 3886 - 3890.

[10] Fernando Morgado Dias, Ana Antunes, Alexandre Manuel Mota. Artificial neural networks: a review of commercial hardware. Engineering Applications of Artificial Intelligence, 2004, vol. 17, issue 8, pp. 945-952A.

[11] B. Dunham, D. Fridshal, R. Fridshal, J. North. Design by Natural Selection. Synthese, D. Reidel Publication Company, Dordrecht-Holland, pp. 254-259, 1963.

[12] Thompson, A., Layzell, P., Zebulum, R. Explorations in Design Space: Unconventional Electronics Design through Artificial Evolution, IEEE Transactions on Evolutionary Computation, vol. 3, № 3, September 1999.

[13] Savage, M.J.W., Salcic, Z., Coghill, G., Covic, G. Extended genetic algorithm for codesign optimization of DSP systems in FPGAs. In Proceedings of IEEE International Conference on Field-Programmable Technology. 2004, pp. 291 - 294.

[14] J. Koza, S. Al-Sakran, L. Jones. Cross-Domain Features of Runs of Genetic Programming Used to Evolve Designs for Analog Circuits, Optical Lens Systems, Controllers, Antennas, Mechanical Systems and Quantum Computer Circuits. NASA/DoD Conference on Evolvable Hardware, IEEE Computer Society Press, 2005, pp. 205 – 214.

[15] G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, M. Fujita. Evolving Robust Gaits with AIBO. In IEEE International Conference on Robotics and Automation, IEEE, 2000, pp. 3040 – 3045.

[16] G. Hornby. Functional Scalability through Generative Representations: the Evolution of Table Designs. Environment and Planning B: Planning and Design, 2004, vol. 31, issue 4, pp. 569 – 587.

[17] Yakymets, N., Kharchenko, V. Resource-Oriented Diversification of Fault-Tolerant PLD-Systems. Radio-electronic and Computer Systems, KhAI, 2006, vol. 3, pp. 45-50.

[18] Sverre Vigander, Evolutionary fault repair of electronics in space applications. Ph.D. Thesis, Dept. of Computer & Information Science, Norwegian University of Science and Technology (NTNU), Trondheim, 2001.

[19] Yakymets, N., Kharchenko, V. Fault-Tolerant Digital Systems Implemented with Partially Definite and Partially Correct Automata. In Proceedings of the 2007 workshop on Engineering fault tolerant systems, Dubrovnik, Croatia, September 04 - 05, 2007.

[20] Yakymets, N., Ushakov, A. Analysis of Failure Types and Development of an Adjustable Generator of Cluster Failures for PLDS. Vestnik NTU KhPI, 2003, vol. 6, pp. 149-152.

[21] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.