Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE APPLYING OF THE HARDWARE-BASED RECONFIGURATION FOR AUTONOMOUS CONTROL SYSTEMS OF SPACE MOBILE ROBOTS

[1]VALERY DMITRIEVICH IVCHENKO, [2]PETR GERMANOVICH KRUG,
[3]MAXIM VYACHESLAVOVICH KURAKOV, [4]EKATERINA NIKOLAEVNA MATYUKHINA,
[5]SERGEY ALEKSANDROVICH PAVELYEV

[1,2,3,4,5]Moscow State Institute of Computer Science
Radio Engineering and Electronics
Prospekt Vernadskogo, 78, Moscow, 119454, Russia

ABSTRACT

The problem of applying of hardware-based reconfiguration for autonomous control systems of space mobile robots is concerned. The challenges associated with using space mobile robots for planetary exploration missions are described and a set of requirements on the design of such space-based robots is imposed. The functional structure of hardware-reconfigurable digital module for intellectual control of mobile space-based robots is proposed and further interaction between its functional modules and remote support center in different situations requiring reconfiguration is concerned. The procedures of self-check and self-testing of the hardware-reconfigurable digital module for intellectual control of mobile space-based robots are described, which are necessary to ensure reliability of reconfiguration. The algorithms for self-testing of the hardware of the digital control module are researched, taking the advantage of on-line partial reconfiguration ability of FPGA. The means to achieve optimal testing coverage while minimizing the amount of additional testing hardware and testing time are considered.

Keywords: *Space-Based Robots, Remote Modification, Mobile Robot, Reconfigurable Computing, Field-Programmable Gate Array (FPGA)*

1. INTRODUCTION

Using space mobile robots for planetary exploration missions is associated with a number of challenges not present when using mobile robots on Earth. First, robots in space have many hardware limitations. The sensible parts have to be protected from dangerous radiations, which usually limits their processing power. In addition, the difference in gravity makes it a lot harder to make good articulations. This usually demands specific design and control procedures. Another more challenging issue is the fact that these robots in space have to operate with no direct human contact. They can receive new instructions and some software upgrades via satellites, but cannot be touched and repaired, and they have to follow the orders autonomously.

Taking this in consideration, as well as long duration of planetary exploration missions, taking months and years, a specific set of requirements on the design of space-based robots is imposed [13], such as:

– High level of survivability – ability to continue to carry out the mission in case of partial malfunction of mechanical parts, hardware or software, as well as in case of unpredictable changes of external environment parameters;

– High level of adaptability (reserve) – ability to continue to work in case of change of target (change of mission) or change of target acquisition way.

Fulfillment of the above-mentioned requirements is possible if autonomous control systems of space mobile robots had the ability to be reconfigured (modified) according to new emerging tasks on strategic or tactical level or the changing circumstances.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

One of the most important problems is to ensure continuous functioning of space-based robots during the whole duration of a mission and to provide possibility of switching to new missions after completion of current tasks by using new configuration of control modules. This problem can be solved with the help of real-time partial reconfiguration - an emerging technology in the field of hardware reconfiguration. Optimal solution would be reconfiguration of specific control modules of space-based robots in background mode without affecting the performance of any modules not taking part in reconfiguration.

In this case, attention should be paid to ensuring the reliability of reconfiguration process. Current technological trends in this field include implementation of automated and automatic means for detection and localization of malfunctions.

Alongside reliable reconfiguration process, it is necessary to ensure reliable transfer of configuration data for remote reconfiguration, as well as to transmit, systematize and store information on reconfiguration and status of control modules of space-based robotic devices in remote support center database.

In the framework of research program "MARS-500" of the Institute of Biomedical Problems of the Russian Academy of Sciences and European Space Agency the mobile robot-explorer "Turist" has been created with the purpose of expanding human capabilities in exploration of aggressive environments, including other planets [12]. Research revealed the disadvantages of the robot-explorer "Turist" as well as other space-based mobile robots requiring manual remote control. Inability of the mobile robot to independently carry out interpretation of events and control operations necessary to reach mission goal was caused by the lack of universal intelligence and corresponding algorithms capable of taking into account all possible missions in advance and all possible situations emerging in the course of planetary missions. Solving this issue would only be possible with the implementation of the technology of autonomous remote modification of hardware.

2. STRUCTURE OF HARDWARE-RECONFIGURABLE DIGITAL MODULE FOR INTELLECTUAL CONTROL OF SPACE-BASED ROBOTS

Considering the above-mentioned tasks, the system for intellectual control of mobile space-based robots based on hardware-reconfigurable digital platform should include the following set of equipment:

1) Reconfiguration controller, utilizing neural network classifier for identification of malfunctions which could occur in the process of remote reconfiguration of the intelligence of space-based robots [5];
2) Reprogrammable hardware on the basis of FPGA, carrying out control on strategic, tactical and operational levels [16];
3) Reconfiguration server, located within remote supervision center, which is connected to space-based robots through the network infrastructure and performs remote control over a group of space-based robots, including coordinated reconfiguration of software and hardware.

Specifications for remote reconfiguration of the intelligence of mobile space-based robots include the following:

1) Automatic remote reconfiguration without operator participation [9];
2) Reconfiguration without physical exchange of hardware;
3) Simplifying the process of technological improvement of the device;
4) Updating the intelligence of space-based robots with state of the art technologies through its remote reconfiguration [10].

Functional specifications for hardware-reconfigurable digital module for intellectual control of mobile space-based robots include the following [19]:

1) Situational control, when each class of possible system states corresponds to a given class of possible solutions;
2) Hierarchical structure of the intellectual control system, including strategic behavior planning level, tactical action planning level, operational (actuator) level and sensors;
3) Justified selection of procedures to run for the solution of a given task on each hierarchical level;
4) Automatic redistribution of tasks between mobile robots within a group of space-based robots.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

System specification for hardware-reconfigurable digital module for intellectual control of mobile space-based robots include the following:

1) Data exchange with other space-based robots working in a group;
2) Reconfiguration of the intelligence should take into account configurations of similar space-based robots working in a group with the purpose of maintaining compatibility and common standards on data exchange [15];
3) Initiation of self-check by remote support center or locally [6];
4) Informing the remote support center about the state of mobile space-based Robot through warning and status messages;
5) Continuous work of mobile space-based robots for the duration of the mission and after reaching the target with the purpose of switching to other missions with the use of new remotely modified intelligence [18].

Based on the above-mentioned specifications the following functional structure of hardware-reconfigurable digital module for intellectual control of mobile space-based robots is proposed (Figure 1).

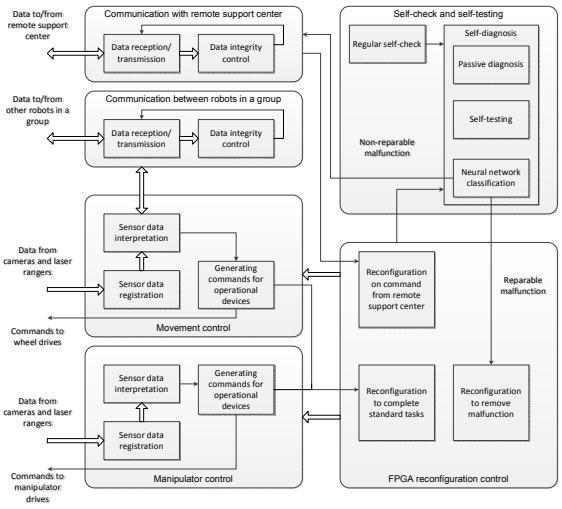

Fig.1. Functional Structure Of Hardware-Reconfigurable Digital Module For Mobile Space-Based Robots Intellectual Control

Digital module for intellectual control of mobile space-based Robot provides communication between mobile robot and remote support center, carried out through standard wireless network interfaces. Space-based robots receive new configuration files from remote support center to perform reconfiguration of FPGA on external command. After reception of new configuration file, the integrity of obtained data is checked using the method of cyclic redundancy check. If integrity check is successful, new configuration is saved in the memory of the digital control module and can be later used for reconfiguration of FPGA. If integrity check fails, obtained data is discarded as corrupted and corresponding message is sent to remote support center. Besides that, the digital module for intellectual control of mobile space-based robot sends to remote support center information about the status of mobile robot and relevant research data obtained during the mission.

Communication between mobile space-based robots within a group is organized similarly. Space-based robots belonging to a single group can exchange status information with each other, which allows them to carry out collective tasks and automatically replace each other in case one or several robots in a group become unable to fulfill their mission.

Information about external environment is obtained by the digital control module with the help of video cameras and laser rangers mounted on the moving platform of space-based robot. Based on this information it issues commands to the moving platform and manipulator arm of space-based robot. Information obtained from sensors is processed in three steps. Initial set of digitalized data is registered in the memory of the digital control module and scaled accordingly for further processing. Next data interpretation is carried out which results in obtaining information about current position of the moving platform and manipulator arm of space-based robot in relation to the goal of the mission and potential obstacles. Results of data interpretation are then used by movement control and manipulator control algorithms to generate commands for operational devices necessary to reach current goal and avoid the obstacles. Movement control is carried out by issuing commands to the drives setting in motion the wheels of the moving platform of space-based robot. Manipulator control is carried out by issuing commands to the drives moving sections of the manipulator arm along their respective degrees of freedom.

Reconfiguration of FPGA can be carried out on command from remote support center or autonomously in response to changes in mission objectives or external environment [15]. Reconfiguration can also be initiated to remove detected malfunctions in the functioning of space-based robot [8].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Self-check of the modules of space-based robot is carried out on initialization of the digital control module and then performed periodically during its runtime. In case self-check discovers a malfunction self-diagnosis procedure is initialized which includes analysis of diagnostic parameters during normal functioning of space-based robot (passive diagnosis) and self-testing based on a set of tests prepared for specific configuration. Determination of malfunction type and its localization is carried out with the help of neural network classifier. In case malfunction can be removed using available hardware resources of FPGA the reconfiguration procedure is initiated. In case malfunction is non-reparable mobile space-based robots stops its current mission and sends corresponding message to remote support center.

Self-check is also performed every time a new configuration is loaded into FPGA. If self-check is successful corresponding control modules of space-based robot switch to running with new configuration. If self-check fails, control modules continue running with the last valid configuration. In case detected malfunction in the new configuration is reparable, another attempt at reconfiguration is performed, otherwise reconfiguration is cancelled and corresponding message is sent to remote support center.

3. FUNCTIONAL PERFORMANCE OF HARDWARE-RECONFIGURABLE DIGITAL MODULE FOR INTELLECTUAL CONTROL OF SPACE-BASED ROBOTS

Interaction between functional modules of the digital module for intellectual control of mobile space-based robots can be described in a set of timing charts.

Figure 2 represents timing charts of standard initialization procedure of the hardware-reconfigurable digital module for intellectual control of mobile space-based robots.

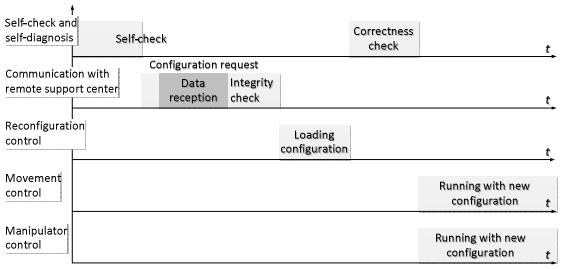

Fig.2. Timing Chart Of Standard Initialization Procedure Of The Digital Control Module

According to standard initialization procedure first self-check of all functional modules is performed, which includes checking the states of error signals reporting general status of each functional module. If self-check is successful a request for configuration file is sent to remote support center accompanied by information about current status of the digital control module.

After sending request for configuration file, the digital control module waits for data from remote support center. Along with configuration file, remote support center sends a set of tests for correctness check after loading the configuration into FPGA. After reception of data, its integrity is checked using the method of cyclic redundancy check. If integrity check is successful, the configuration file is loaded into FPGA.

After loading the configuration into FPGA its correctness is checked using the corresponding set of tests. If correctness check is successful movement, control and manipulator control modules of space-based robots switch to running with new configuration. At this point, initialization procedure is considered complete and the digital control module switches to standard mission performance mode.

During standard mission performance in case space-based robot receives new mission goals, the means of reaching the goal have to be changed or under other similar conditions FPGA configurations of movement control and manipulator control modules can be changed on command from remote support center. Figure 3 represents timing charts of FPGA reconfiguration procedure on command from remote support center.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

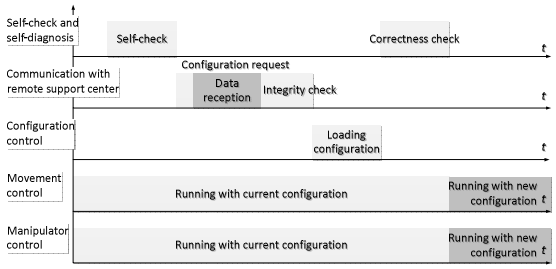

Fig.3. Timing Chart Of FPGA Reconfiguration Procedure On Command From Remote Support Center

During standard mission performance, communication between space-based robot and remote support center is carried out periodically in set time frames. Before communication is initiated self-check of all functional modules of space-based robot is performed. If self-check is successful, a request for configuration file is sent to remote support center accompanied by information about current status of the digital control module and research data obtained during the mission if it is included in current tasks of the mobile space-based robot.

After sending request for configuration file, the digital control module waits for data from remote support center. If remote support center sends a new configuration file it should be accompanied by, corresponding command which signals the digital control module that it needs to run reconfiguration procedure. Along with configuration file, remote support center sends a set of tests for correctness check after loading the configuration into FPGA. After reception of data, its integrity is checked using the method of cyclic redundancy check. If integrity check is successful, the configuration file is loaded into FPGA.

After loading the configuration into FPGA its correctness is checked using the corresponding set of tests. If correctness check is successful movement, control and manipulator control modules of space-based robots switch to running with new configuration. Presented time charts depict reconfiguration of both movement control and manipulator control modules but each module can also be reconfigured separately. Reconfiguration procedure does not affect current performance of the mobile space-based robot until the moment corresponding module switches to running with new configuration.

Besides commands from remote support, center reconfiguration procedure can be initiated locally on fulfillment of specific conditions. For example if a malfunction is detected in FPGA of movement control or manipulator control modules the performance of space-based robot can be restored by loading configuration carrying the same functionality but utilizing spare hardware resources not affected by the malfunction. Figure 4 represents timing charts of FPGA reconfiguration procedure on detection of a malfunction.

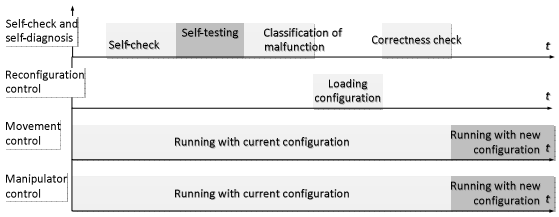

Fig. 4 Timing Chart Of FPGA Reconfiguration Procedure On Detection Of A Malfunction

If periodical self-check detects a malfunction in FPGA of movement control or manipulator control modules self-testing procedure is carried out which localizes the malfunction and collects data for its classification. According to results of self-testing classification of malfunction is carried out. Based on classification results a decision is made whether malfunction can be removed by performing reconfiguration procedure using one of configuration files stored in the memory the digital control module.

After loading the configuration into FPGA standard correctness check is carried out. If correctness check is successful movement, control and manipulator control modules of space-based robots switch to running with new configuration.

4. SELF-CHECK AND SELF-TESTING OF HARDWARE-RECONFIGURABLE DIGITAL MODULE FOR INTELLECTUAL CONTROL OF SPACE-BASED ROBOTS

Self-check and self-testing are parts of automated control system monitoring the status of all modules of space-based robot, which is meant to detect, localize and classify malfunctions appearing in space-based robot, including malfunctions in the hardware-reconfigurable digital control module. Based on classification results detected malfunctions can be removed automatically by performing reconfiguration procedure.

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Automated control system monitoring the status of all modules of space-based robot has hierarchical structure. Self-check constitutes the first and the lowest control level. Self-check includes a set of short procedures implemented in hardware and not affected by any changes in FPGA configuration. Malfunction in any module of space-based robot leads to generation of an error signal which is registered during self-check. Main purpose of self-check is preventing the functioning of space-based robot with hardware malfunction, which could lead to further damage and bringing the mobile robot out of commission. Self-check is carried out on initial start-up of space-based robot and then periodically in set of time frames.

Figure 5 represents the algorithm of self-check of hardware-reconfigurable digital module for intellectual control of space-based robots.

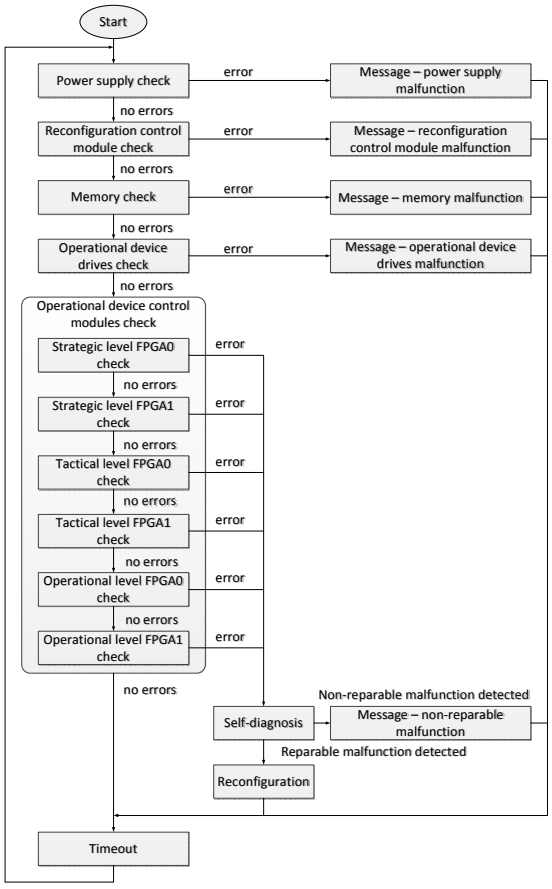

Fig.5. Algorithm of self-check of the digital control module

Self-check procedure includes the following steps carried out sequentially:

1) Self-check initialization.
2) Power supply check - no errors signal means that all output voltages satisfy working conditions, otherwise a message about power supply malfunction is sent to remote support center.
3) Reconfiguration control module check - no errors signal means that a valid configuration is loaded in reconfiguration control module and all necessary clock frequencies are being generated, otherwise a message about reconfiguration control module malfunction is sent to remote support center.
4) Memory check - no errors signal means that all memory blocks successfully passed write and read checks, otherwise a message about memory malfunction is sent to remote support center.
5) Operational device drives check - no errors signal means that all operational device drives are ready to receive commands from control modules, otherwise a message about operational device drives malfunction is sent to remote support center.
6) Operational device control check - is divided into steps checking FPGA responsible to control over strategic, tactical and operational levels. Two FPGA are functioning on each level - active and reserve. Active FPGA receives data, performs data interpretation and generates commands according to current configuration. Reserve FPGA is used to load new configuration and receives control after successful correctness check, at the same time previously active FPGA switches to reserve mode. No errors signal means that a valid configuration is loaded in active FPGA and all necessary clock frequencies are being generated on both FPGA, otherwise self-diagnosis procedure is initialized for malfunctioning FPGA. Based on results of self-diagnosis detected malfunction can be classified as reparable or non-reparable. In case or reparable malfunction FPGA reconfiguration procedure is initialized to remove it, otherwise a message about non-reparable malfunction in operational device control module is sent to remote support center.
7) After conclusion of all self-check steps, timeout of a set duration takes place after which self-check procedure is initialized again.

Self-check results only indicate the presence of a malfunction in corresponding module of mobile space-based robot without showing its exact

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

location. Self-diagnosis procedure performs localization and classification of malfunctions within hardware-reconfigurable digital module for intellectual control of space-based robots. Self-diagnosis of hardware-reconfigurable digital control module includes passive diagnosis, self-testing and classification of malfunction type with the help of artificial neural network.

Passive diagnosis is carried out by hardware control procedures implemented in current configuration and working independently from the main task performed by FPGA. Hardware control can detect errors in functioning of FPGA in working mode without usage of special tests. However, using only hardware control leads to significant complication of configuration and can not always provide full coverage. Including large amounts of complicated additional logic in a configuration can also reduce its overall reliability. Because of this passive diagnosis is used together with self-testing for initial localization of malfunctions or exclusively in case self-testing is impossible.

Tests for FPGA are divided into two groups: manufacturer tests and user tests [3], [22]. User tests can also be divided into two types: 1) system integrator tests [2], [7], [21] and 2) field tests [1], [11], [20]. In the field devices carrying out crucial tasks (such as space devices) demand fast testing of hardware with minimal interruption of functioning of the device as a whole.

In self-testing mode FPGA is reconfigured in such a way, that configuration under testing is decomposed into basic elements which are represented by configurable logic blocks of FPGA. To do this all connections of each block with external structure are cut. Then all blocks are checked with an exhaustive test (full coverage test [3]. Test signals are forwarded to all inputs of FPGA and connection nodes from test generator and output signals from each block are received by malfunction classifier. Both test generator and malfunction classifier can be implemented within the same FPGA using available hardware resources. As a result, simultaneous testing of all blocks and all connections is carried out. Connection here is defined as plurality of all metal traces, switching matrices and pass transistors taking part in forming a given connection. Pass transistor is a special connection transistor controlled by a block of configuration memory. In a logical matrix it makes up a programmable interconnect point (PIP).

Connecting test generator to block outputs instead of block inputs not only allows simultaneous testing of connections together with blocks but also significantly lowers the amount of necessary connections between test generator and tested object. Instead of M*N connections, where M - number of block inputs, and N - number of blocks in a matrix, required number of connection amounts to N+Nin, where Nin - number of external inputs of tested object.

To forward test signals to all blocks the outputs of test generator should be connected to the nodes Gi, i = 1, ... ,N formed after decomposition of the configuration in such a way that all inputs of each block received signals from different order of test generator output. Such connection is called correct. Only correct connection ensures full coverage of possible inputs for all blocks. Giving a node, a certain order number of test generator output is called assignment. Assignment task in the proposed architecture is not trivial since it is complicated by branching nodes. This means that assigning a number to a block input which is a branch of a branching node automatically assigns the same number to all other branches of this node which can lead to contradictions among inputs of other blocks. Contradictions make correct assignment for this block impossible (in a way that all inputs of the block had different numbers). This problem can be solved with the following statement.

Statement 1. For every configuration, a correct assignment can always be found by using test generator with larger order number.

Increasing order number by one leads to increasing test length two times. Upper limit for the necessary increase of order number can be estimated by reduction of assignment task to the known task of coloring in graph theory [14]. Tree edge coloring cannot be implemented here because in these task branches of a branching node are represented by different tree edges belonging to the same tree node and such edges should be assigned different colors. In the task in question, such edges should be assigned same colors (having the same assigned number). Tree node coloring task should be used instead, and the graph for the configuration where assignment is taking place should be built according to following rules (such graph is called converse):

– Each node Gi, i = 1, . . . , N, of the initial configuration is assigned to a node of the converse graph;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

– Between two nodes Gi and Gj of the converse graph an edge is formed if in initial configuration there is at least one block which is incidental to both nodes Gi and Gj .

Since in initial configuration every node is a block output (branching or non-branching) or an external input (branching or non-branching) total number of nodes B0 in the converse graph is equal to

$$B_0 = N + Nin \quad (1)$$

(Where N – number of blocks, Nin – number of external inputs).

As is known [20], chromatic number [gamma] estimated through the number of nodes of the graph lies in the following range:

$$1 \le \gamma \le B_0 \quad (2)$$

This shows that assignment task can theoretically lead to unacceptable increase of order number of the test generator (for complicated heavily branching configurations) due to timing constraints. This problem can be solved with the following statement.

Statement 2. For every configuration, a correct assignment can always be found by introducing additional nodes through connection cuts.

Using statement 2 brings an additional degree of freedom to the configuration allowing assignment of numbers to the branches of a branching node independently from each other. As a result, it allows avoiding increase the order number of the test generator.

In a general case, finding the optimal solution for introducing additional nodes is a sequential search task. For its solution, an assignment algorithm [17] is proposed utilizing statements 1 and 2. Algorithm is based on the following ideas.

Assignment starts from the nodes with the most branches. Otherwise, such nodes would have little freedom for assignment since they are connected to a large number of previously assigned block inputs.

Non-branching nodes are assigned last since there will always be a free number to assign to them which was not previously used for this block.

First block to cut connection with should be the one having an input with assigned number, which is not assigned to any other blocks connected to the same branching node. This number is assigned to disconnected branching node and the (one) cut block will always have a non-contradictory number.

If there is no such singular block first blocks to cut connection with should be the ones with a number less frequently met (lowest entry number) since in such case it is likely less connection cuts will be needed. If there are several such options the blocks with the same, missing number should be chosen. For example if connections are cut with two blocks with assigned numbers (5, Gi, 4, 2) and (4, 3, 5, Gi) the missing number for both blocks is 1. Then the branching node Gi is assigned number 1.

Assignment algorithm includes the following steps:

Step 1. Any node with the most branches, which is not yet assigned, is chosen. This node is assigned number 1, which determines the assignment n = 1 to all its branches.

Step 2. Next non-assigned branching node is chosen, if it exists (otherwise go to step 4). This node and all its branches are assigned n = 1. A check is made if there is an incidental block, which received the same assignment to two inputs:

2a. If there are no such blocks then step 2 is repeated.

2б. If there is such block then assigned number is increased by one: N = N + 1 (i.e. every time assignment attempts for the chosen node start from number 1 and end with assignment of the lowest number for which no incidental blocks have a pair of inputs assigned the same number). Assigned number is compared to the given maximum number Mgt for the order of test generator output:

If n ≤ Mgt, then n is assigned to the chosen node and step 2 is repeated.

If n > Mgt, then go to step 3.

Step 3. Among incidental, blocks for the chosen node the ones with the lowest entry number are determined. Determined blocks are disconnected using a switch, which makes a new independent node Gn+1. The previously chosen node is assigned the lowest entry number and the new node Gn+1 is assigned order according to the algorithm. The number of nodes in the configuration now increased by one: N = N + 1. Go to step 2.

Step 4. In an arbitrary order blocks with non-assigned inputs are determined (i.e. search for non-branching nodes is performed). Such inputs are assigned numbers not coinciding with already assigned inputs of the same block. Algorithm ends when all inputs of all blocks are assigned.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Results of assignment algorithm are used for routing test generator outputs to the object of testing.

Figure 6 represents the algorithm of self-testing of hardware-reconfigurable digital module for intellectual control of space-based robots.

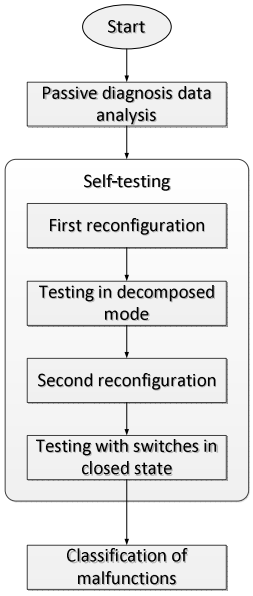

Fig.6. Algorithm Of Self-Testing Of The Digital Control Module

Self-testing of the object implemented on a programmable logic matrix structure consists of several stages.

Stage 1. First reconfiguration. FPGA is loaded with configuration in decomposed structure described above. To do this PIP corresponding to switches on outputs of every block are opened and PIP connecting the test generator and malfunction classifier to resulting nodes are closed.

It is worth mentioning that each switch should disconnect all branches from a given block output at once. Thus during routing process a switch should be put on the output of each block (and each external input) which would be common for all following branches (i.e. located before the first branching node).

Stage 2. Testing of the object in decomposed mode. First configuration is tested in the following way: test generator forwards an exhaustive test to all logical blocks; output reaction is registered by malfunction classifier; malfunction classifier analyzes obtained data by comparison with the reference signature of the object and generates corresponding signal about technical state of the object.

Stage 3. Second reconfiguration. FPGA is partially reconfigured in the following way: switches are closed; test generator and malfunction classifier remain connected. In this configuration, closed switches are tested since they remain in such state during standard functioning of the tested object.

Stage 4. Testing with switches in closed state. Such test requires only two sets of data from the test generator (all zeros and all ones, or any set and its inversion) which can then be analyzed by malfunction classifier. But for that all block outputs should remain in high impedance mode. If specific logic matrix can not provide it, the problem can be solved through switching means. For example, another switch can be sequentially connected to switches on block outputs. In the first configuration, such switch would be closed and its state is tested on stage 2. In the second configuration, this switch is opened and removes tested switch on block output from the output signal of that block. This allows testing of switches on the outputs of each block: signal from the test generator passes through the tested switch in direction opposite to the one during standard functioning of the tested object and is registered by malfunction classifier.

To reduce the load on the test generator outputs and shorten the length of connections between the test generator and the tested object it is proposed to implement several test generators in the logical matrix, each providing tests for its specific part. Results of passive diagnosis can be used to determine the parts of FPGA configuration, which require testing.

5. RESULTS

Functional structure of hardware-reconfigurable digital module for mobile space-based robots intellectual control is designed with capability for on-demand reconfiguration of hardware, which may be required due to switching to another mission, acquisition of new tasks on strategic or tactical level or changes in external environment. Reliability of reconfiguration is ensured through multi-step process of self-check and self-testing, taking the advantage of on-line partial reconfiguration ability of FPGA.

Implementation of the proposed algorithm of self-testing of hardware-reconfigurable digital module for intellectual control of space-based

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

robots provides 100% detection of all standard malfunctions (single and multiple errors, constant and logical errors, short circuits and connection losses, but not the ones multiplying the number of logical block inputs) and almost all malfunctions of non-standard types (i.e. multiplying the number of logical block inputs). Using exhaustive testing makes it not necessary to design specialized testing sequences. Including the testing hardware into FPGA configuration does not bring additional latency to existing nets. Parallel testing of logical blocks and relatively short length of exhaustive test result in lower testing time. Additionally the complexity of testing hardware does not depend on complexity of the tested object itself.

Testing of the hardware-reconfigurable digital module for intellectual control of mobile space-based robots was carried out using seven sets of configuration files which included different types of malfunctions arranged arbitrarily: single constant error in a configuration; multiple constant errors; single logical error in a configuration; multiple logical errors; single short circuit or connection loss in a configuration; multiple short circuits or connection losses; multiplication of logical block inputs. Figure 7 represents the results of detection of malfunctions and identification of reparable malfunctions by the classifier.

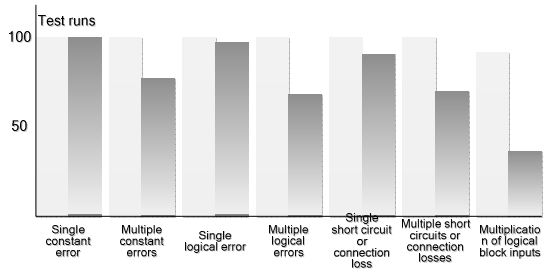

Fig.7. Detection Of Malfunctions And Identification Of Reparable Malfunctions

Achieved results show that implementation of hardware-reconfigurable digital module for mobile space-based robots intellectual control with the proposed functionality of self-testing allows the detection of almost all hardware malfunctions that may occur during runtime or reconfiguration process and in significant amount of cases autonomous repair with the use of reconfiguration ability can be carried out.

6. CONCLUSION

Proposed structure of hardware-reconfigurable digital module for intellectual control of space-based robots is designed to improve survivability and self-sufficiency of mobile space-based robots with the help of technologies of remote modification of intelligence based on reconfigurable hardware. Developed concept brings together the methods and technologies of artificial intelligence and capabilities of real time partial reconfiguration of modern FPGA as the means to adapt to unpredictability of external environment. Practical implementation of this concept proposes selective usage of most suitable knowledge processing technologies depending on specific task, properties of the object of control, its functional purpose, working conditions etc. Implementation of the developed technologies will contribute to the efficiency of space-based robots performance and thus to the reduction of space mission costs.

The problem of utmost importance is the reliability of dynamically reconfigurable autonomous control system, which gives prospect to further development of self-check and self-testing procedures of hardware-reconfigurable digital module for intellectual control of space-based robots. In the nearest future research on development of intellectual malfunction classifier, which could allow application of partial reconfiguration of FPGA to restore malfunctioning configuration using free hardware resources is being envisioned.

Practical implementation of the proposed hardware-reconfigurable digital module for intellectual control of space-based robots is being developed in the framework of further improvement of the space-based mobile robot-explorer "Turist" carried out by the Moscow State Institute of Computer Science Radio Engineering and Electronics. Implementation of autonomous remote modification of hardware would improve survivability and allow the mobile robot to continue its work in case of change of environment or mission goals. For this purpose the proposed technologies are being adapted to the use with specific set of on-board equipment.

7. ACKNOWLEDGEMENT

The research was carried out with financial support of the Ministry of Education and Science of the Russian Federation in the framework of the Agreement # 14.574.21.0102, 31.07.2014

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1]. Aitken J., Veres S., Judge M. (2014). Adaptation of System Configuration under the Robot Operating System. Proceedings of the 19th IFAC World Congress (pp. 4484-4492). Cape Town, South Africa: Cape Town International Convention Centre. http://dx.doi.org/10.3182/20140824-6-za-1003.02531.

[2]. Bolchini C., Miele A., Santambrogio M.D. (2007). TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs. Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (pp. 87-95). Rome, Italy. http://dx.doi.org/10.1109/dft.2007.25.

[3]. Brooks A., Kaupp T., Makarenko A., Williams S., Oreback A. (2005). Towards component-based robotics. Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 163-168). http://dx.doi.org/10.1109/iros.2005.1545523.

[4]. Brooks R. (1989). A robot that walks; emergent behaviors from a carefully evolved network. Proceedings of the 1989 International Conference on Robotics and Automation (pp. 692-696 vol. 2). Scottsdale, AZ, USA. http://dx.doi.org/10.1109/robot.1989.100065.

[5]. Commuri S., Tadigotla V., Sliger L. (2007). Task-based Hardware Reconfiguration in Mobile Robots Using FPGAs. Journal of Intelligent and Robotic Systems, 49 (2), 111-134. http://dx.doi.org/10.1007/s10846-007-9131-3.

[6]. Das S.R. (2005). Self-testing of cores-based embedded systems with built-in hardware. IEE Proceedings - Circuits, Devices and Systems, 152 (5), 539-546. http://dx.doi.org/10.1049/ip-cds:20045050.

[7]. Duarte C., Martel G., Buzzel C. (2005). A Common Control Language to Support Multiple Cooperating AUVs. Proceedings of the 14th International Symposium on Unmanned Untethered Submersible Technology. Lee, New Hampshire, USA: Autonomous Undersea Systems Institute.

[8]. Dutt S., Verma V., Suthar V. (2008). Built-in-self-test of FPGAs with provable diagnosabilities and high diagnostic coverage with application to online testing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27 (2), 309-326. http://dx.doi.org/10.1109/TCAD.2007.906992.

[9]. Giger G., Kandemir M., Dzielski J. (2008). Graphical Mission Specification and Partitioning for Unmanned Underwater Vehicles. Journal of Software (JSW), 3 (7), 42-54. http://dx.doi.org/10.4304/jsw.3.7.42-54.

[10]. Gokhale M., Graham P., Wirthlin M., Johnson D.E., Rollins N. (2006). Dynamic reconfiguration for management of radiation-induced faults in FPGAs. International Journal of Embedded Systems, 2 (1/2), 28-38. http://dx.doi.org/10.1504/ijes.2006.010162.

[11]. Hernandez C., Bermejo-Alonso J., Lopez I., Sanz R. (2013). Three Patterns for Autonomous Robot Control Architecting. Proceedings of the Fifth International Conferences on Pervasive Patterns and Applications PATTERNS 2013 (pp. 44-51). Valencia, Spain.

[12]. Ivchenko V., Krug P., Matyukhina E., Pavelyev S. (2015). The Mars-500 Program Space-Based Mobile Robot "Turist". Applied Mechanics and Materials, 789-790, 742-746. Doi: 10.4028/www.scientific.net/AMM.789-790.742.

[13]. Ivchenko V., Krug P., Morozova T., Ostroukh A., Pavelyev S. (2014). The Remotely Reconfigurable Intelligence of the Space-Based Mobile Robot. Journal of Engineering and Applied Sciences, 9 (10), 389-395. Retrieved from http://medwelljournals.com/abstract/?doi=jeasci.2014.389.395.

[14]. Kleinehagenbrock M., Fritsch J., Sagerer G. (2004). Supporting advanced interaction capabilities on a mobile robot with a flexible control system. Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3649-3655 vol. 4). Sendai, Japan. http://dx.doi.org/10.1109/iros.2004.1389982.

[15]. Li L., Chakrabarty K., Kajihara S., Swaminathan S. (2005). Three-stage compression approach to reduce test data volume and testing time for IP cores in SoCs. IEE Proceedings - Computers and Digital Techniques, 152 (6), 704-712. http://dx.doi.org/10.1049/ip-cdt:20045150.

[16]. Menon P., Xu W., Tessier R. (2006). Design-specific path delay testing in lookup-table-based FPGAs. IEEE Transactions on Computer-Aided Design of Integrated Circuits

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and Systems, 25 (5), 867-877. http://dx.doi.org/10.1109/tcad.2005.855955.

[17]. Merz T. (2004). Building a system for autonomous aerial robotics research. Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles. Amsterdam, Netherlands: Elsevier.

[18]. Merz T., Rudol P., Wzorek M. (2006). Control System Framework for Autonomous Robots Based on Extended State Machines. Proceedings of the 2006 International Conference on Autonomic and Autonomous Systems (p. 14). Silicon Valley, CA, USA. http://dx.doi.org/10.1109/icas.2006.19.

[19]. Moubarak P., Ben-Tzvi P. (2012). Modular and Reconfigurable Mobile Robotics. Journal of Robotics and Autonomous Systems, 60 (12), 1648–1663. http://dx.doi.org/10.1016/j.robot.2012.09.002.

[20]. Patil M, Abukhalil T., Sobh T. (2013). Hardware Architecture Review of Swarm Robotics System: Self-Reconfigurability, Self-Reassembly, and Self-Replication. ISRN Robotics, 2013, 1-11. http://dx.doi.org/10.5402/2013/849606.

[21]. Sedcole P., Blodget B., Becker T., Anderson J., Lysaght P. (2006). Modular dynamic reconfiguration in Virtex FPGAs. IEE Proceedings - Computers and Digital Techniques, 153 (3), 157-164. http://dx.doi.org/10.1049/ip-cdt:20050176.

[22]. Sutton R. S., Barto A. G. (1998). Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press. http://dx.doi.org/10.1016/s0893-6080(99)00098-2.