Reconfigurable NoC Development with Fault Mitigation

Elena Suvorova, Yuriy Sheynin, Nadezhda Matveeva
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation
{suvorova, sheynin}@aanet.ru, nadezhda.matveeva@guap.ru

***Abstract*—An ability of faults mitigation becomes one of the main requirements for network-on-chip (NoC) embedded systems that are manufactured with thin design rules. Other requirements are for area, power consumption and QoS. Altogether these requirements lead to inconsistencies in NoC development process. In this paper we discuss approaches for resolving this conflict by the use of the dynamically reconfigurable NoC. We suggest a methodology for development of NoC with fault mitigation support. The methodology allows to take into account specifics of design process and operating conditions in the implementation of a particular project. For verification of the suggested approach we use analysis and simulation. We suggest an approach to simulation that enables dynamic failure injection.**

I. INTRODUCTION

A modern system-on-chip (SoC) for different applications should be resistant to failures. These requirements are relevant both for developed for SoC, developed for embedded systems with thin design rules. Failures and accelerated aging in for embedded systems NoCs that are developed with thin design rules (currently from 40 nm to 14 nm) occur due the specific of a thin technology.

In this paper we consider existing failure mitigation approaches for NoC, evaluate its advantages and disadvantages for embedded systems with thin design rules. We suggest the methodology for development NoC with fault mitigation.

In the 2nd section we consider causes of faults and failures in the SoC and their main types. In the 3rd section we consider the fault mitigation techniques in circuits. In the 4th section we consider faults mitigation techniques in NoC structure. In the 5th section we represent our method of reconfigurable NoC structure organization with support of fault mitigation. In the 6th section we represent results of analysis and simulation of developed by our method NoCs.

II. CAUSES OF FAULTS AND FAILURES

A. *Causes of faulst and failures in SoCs*

The main types of errors that are caused:

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

The correct functionality of the SoC could be restored after a soft error. The hard errors occur as a result of irreparable damage.

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.

Hard errors are:

- Stuck at fault (‘0’, ‘1’) – errors when value in one point of circuit has constant value “0” or “1”. It happens due to physical destruction of floorplan such as gap lane.
- Single event latch-up (SEL) – dramatically increases the leakage current;

B. *Causes of faults and failures in a SoC for embedded systems manufactured by thin design rules*

The main cause of faults and failures in a manufactured by thin design rules SoC for embedded systems (40 nm and below) are technology features. First of all it is the technology vulnerability (process variability) [1], [2]. The technology vulnerability can occur both within one wafer and within different wafers. Details of the topology of each chip are slightly different from each other. Therefore similar circuit elements have a different shape in different chips. The actual shape differs from rectangular. It can have rough edges. In some cases it has trapezoid instead of the rectangular shape that was planned.

These differences are negligible small in most of manufactured by large design rules chips. The chips with significant differences are rejected during the manufacturing tests (amount of it usually is less than 1 – 3%). For thin design rules these differences are more significant, therefore each chip is unique. As a result, just rejection of chips in not applicable, because it leads to a very low yield (less than 50%). Stuck at fault (‘0’, ‘1’), SEL can be at a chip due to manufacturing defects. Also there are hidden faults which lead to SEU, MCU, SET. Chips with SEL are rejected on the step

of manufacturing testing. As a consequence different fault mitigation techniques should be integrated into a chip.

Technology vulnerability can lead to hard errors, such as line breakdown, and to soft errors, for example, due to reducing a distance between adjacent lines, a change of signal value in one line could induce a glitch in another. When flip-flop is made with thin design technology, the triggering time is very small. It is part of picosecond. A glitch may be enough for triggering of incorrect value. Therefore glitches may be source of SET, SEU. Also they cause MCU when glitch effects on several adjacent lines at the same time [1], [2], [3].

Another problem of the thin design rules is accelerated aging (aging for design rule 40 nm is 3 – 5 times faster than aging for design rule 65 nm) [1], [2]. It is a result of the diffusion process, consequences of which for the design rules 40 nm and below cannot be neglected. Accelerated aging at initial stages leads to soft errors, such as crosstalk in the interconnection lines, and then to hard errors, such as bridges between communication lines.

C. The main types of faults and failures in SoC

TABLE I. THE MAIN TYPES AND CAUSES OF FAULTS

Error type	Thin design rules
Stuck at 0	Technology vulnerability
Stuck at 1	Technology vulnerability
Increasing the switching time of a gate	Accelerated aging
Line breakdown	Technology vulnerability
Bridge between lines	Technology vulnerability; Accelerated aging
Glitch in an interconnection line	Technology vulnerability; Accelerated aging

The main types and causes of faults and failures, which are typically appear in SoC which manufactured by thin design rules for embedded systems represented in Table I. As can be seen from this table, errors may occur in components (flip-flops, logic gates) and in interconnection lines.

III. THE APPROACHES TO FAULT MITIGATION IN CIRCUITS

A. The layers of fault mitigation

The faults mitigation may be implemented at various levels:

- on the level of architecture and structure of NoC;
- on the level of the separate IP-blocks;
- on the level of technology library.

Fault mitigation techniques corresponding to different levels can be used simultaneously.

In this paper we will focus on the fault mitigation techniques implemented on the NoC structure level: communications between components (point-to-point communications), NoC interconnection graphs, basic NoC elements structure.

A NoC should include mechanisms for error detection, error correction mechanisms and/or blocking of faulty components by reconfiguration mechanisms. We consider mechanisms for error detection in interconnection lines, since the choice of concrete mechanism has a significant influence on the structure of switches, routers. (For example, if CRC based mechanism is selected, and a check is performed at data link layer, every port of switch/router should include a buffer.)

B. Fault mitigation in interconnection lines

In [3] the research of different approaches to faults mitigation in interconnection lines was performed and two main groups of approaches were considered: the approaches based on a retransmission of data and the approaches based on a coding. The approaches based on retransmission are more effective than the approaches based on coding from the viewpoint of energy if a retransmission is performed within a single data link. If retransmission is performed between a source and a receiver nodes, for large network size (mean path length over 8 – 10 hops) power consumption for retransmission based approaches is essentially bigger.

The approaches based on retransmission between source and destination nodes have another disadvantage. If the packet header is corrupted due a fault, the packet will be incorrectly routed and consequently deadlock or livelock may occur in the network. Therefore, the developer should include in the NoC additional mechanisms for deadlock and livelocks avoidance. Implementation of the mechanisms can lead to essential overheads.

In [3] is presented the comparison of various coding strategies for protecting data such as parity bits, CRC, Hamming codes. In point of view of hardware costs and time parameters, the CRC mechanism is preferred for NoC. The balance between overhead and the number of identified errors is the best.

However, this mechanism has an essential disadvantage: the buffers, which size corresponds to the size of data object protected by CRC, required for implementation of this mechanism. Therefore, in case when use of buffers is not possible/not desired or the data transmission “on the fly” is required, the mechanism of parity bits is preferably. This coding does not guarantee detection of multiple errors; therefore it is suitable for systems in which the probability of crosstalk is low.

The crosstalk problem is actual for NoCs implemented by thin design rules. Specialized coding may be used to eliminate the effect of crosstalk on interconnection lines. The basic idea of this coding – to eliminate situations such as transfer of “010” after “101” – a large number of inverted values of bits in the serial words. Hamming codes are an example of such coding. However there are special Crosstalk avoidance coding (CAC), they are more efficient than Hamming codes, but hardware cost of its implementation is bigger.

The data protection strategies based on coding methods cannot provide protection against failures. Therefore, combined techniques based on temporal and spatial

redundancy are used [3]. An example of such implementation of data link is shown in the Fig. 1.

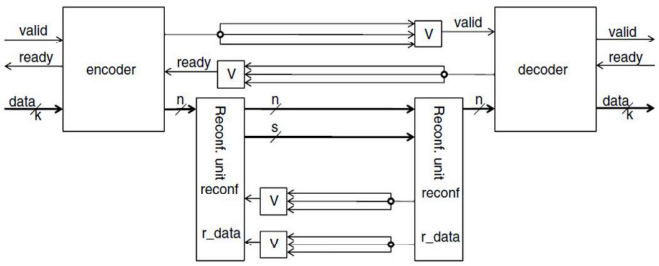

Fig. 1. The example of data link implementation

In this example, triple redundancy is used for valid and ready lines. The data are transmitted via *n* (64) bit lines, every 8 lines (one byte) are followed by one parity line. The spare data lines (*s* bits) can be used when primary data lines fail [3, 4]. Some approaches provide possibility to transmit data "parts" if the number of remaining lines less than *n* [3].

In another approach the set of data (bit) lines is divided in groups. Every group is followed by independent control signals [5].

C. *Fault mitigation in components*

The following main groups of approaches could be distinguished. The first group is based on a duplication of components. The majority circuits may be used. They can include three or even five (for critical blocks) instances of the component and a comparison unit, which allows to obtain at the output a correct result (in case of an error in one or two blocks respectively).

The main disadvantage of this approach is dramatically increasing of SoC area (the probability of SEU, SET, MCU is proportional to the SoC area) and power consumption. For a majority circuits the probability of error increases by N times, where N – number if instances of the block.

TABLE II. FAULT MITIGATION IN COMPONENTS

NoC for embedded systems	40 nm	32 nm, 28 nm	14 nm
Interconnection lines	Sometimes coding for long lines	Coding for long lines	Coding for long and middle lines
Flip-flops (SRAM sells)	The Hamming codes (or others) for configuration and program memory	The Hamming codes (or others) for configuration and program memory, Buffers, Selective protection for FSM	The Hamming codes (or others) for configuration and program memory, Buffers, Selective protection for FSM
Logic gates	Fault mitigation not used	Selective fault mitigation for circuits operation at high frequencies	Selective fault mitigation

For embedded applications increase in energy consumption is most critical. On one hand, it reduced the consumer characteristics; on the other hand the associated additional heating of circuit leads to aging process acceleration. Therefore, majority circuits can be used only for the most critical components of a SoC.

IV. THE APPROACHES FOR FAULT MITIGATION IN NOC STRUCTURE

Most of modern approaches to NoC with faults mitigation development are based on a reconfiguration of interconnection structure and redirection of data flows to bypass failed components and communication lines. The NoC may include a number of complementary interconnections. In addition, data streams can be redirected at the others primary interconnections to bypass the failed components. However, in this case the timing parameters of the interconnection system may decrease due to increase of primary interconnections' load.

Often the efficiency of these approaches depends not only on the quantity of additional communication lines, but also on their relative position. If two lines are placed close to each other possibility of simultaneous failure can be much higher than if they are located far from one to another. Let's consider some typical examples of modern NoCs with fault mitigation on the structure layer.

A bypass path (or multiple backup paths) can be built for the NoC. The path allows data transmission if one of routers or interconnections fails, the example of such path is shown of Fig. 2.

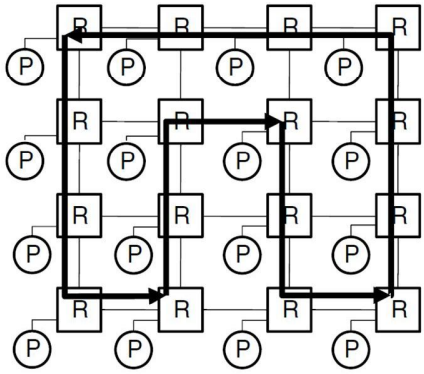

Fig. 2. The example of bypass path

Hardware cost of this approach is very small, but in case of failure in the link between a terminal node and the router, the communication with this terminal node would be lost [3].

In [3] a set of approaches is described, in which terminal nodes may be connected to some routers. The interconnection between the terminal node and an only one of the routers is activated during normal operation. If this router fails, the system may be reconfigured – the terminal node is switched to other router. Various strategies of primary and additional interconnection lines placement can be used in these approaches. In some strategies these interconnection lines are placed "as far as possible" from each other to eliminate simultaneous faults in primary and additional interconnections

caused by a single source. But this strategy can have a negative impact on the length of wires, the number of vias, on data transmission time.

In other strategies the primary line connects the terminal node with one router. In the router the main path (via router) and a bypass are implemented. The bypass is activated when router fails. The example of such implementation is represented on the Fig. 3

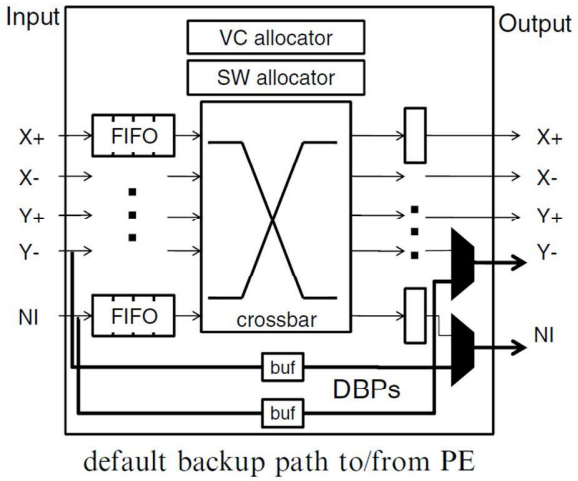

Fig. 3. The example of the router with bypass path

In this example the NI of the terminal node is connected to the router. In this router the bypass path between NI and the output Y-is implemented [6].

In other approaches spare routers that may be used in NoC in case of a primary router failure, is suggested [7].

The Fig. 4 shows an implemented with using this approach NoC. The spare router is added in every column of the grid. Each node is connected to multiple routers via the external multiplexing block. In case of failure in one router the terminal node may send data via others.

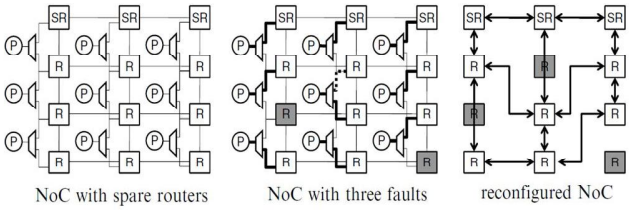

Fig. 4. The example of NoC with spare routers in every column of the grid

In [8] the authors propose to divide the interconnection graph into subgraphs of equal size. In each subgraph one spare router is included, Fig. 5. It can replace any of the primary routers.

In this approach the original interconnection graph may be transformed from regular to irregular structure. This possibility should be taken into account at the level of used in the system routing algorithms.

In other approaches the router is divided into parts. In case of failure in one part, other parts can continue to work. The most critical components of the router typically are duplicated [9].

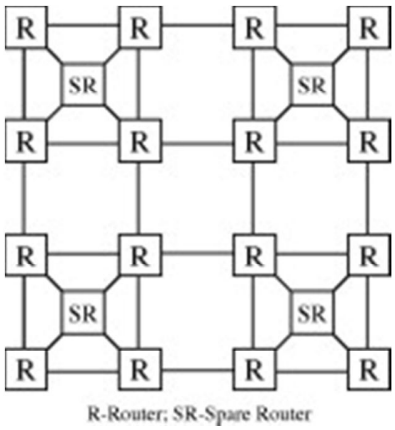

Fig. 5. The example of NoC with spare routers in every group of the routers

In [10] the complex approach to ensure the NoC survivability is represented. In this approach CRC in used to detect faults. BIST is used to identify a place of the fault occurrence. The system includes the tools for bypassing the faulty components. The Fig. 6 show the structure of router used in such a network. Port swapper determines what interconnection line will be connected with which FIFO. If any input port of the router fails, the port swapper will take into account.

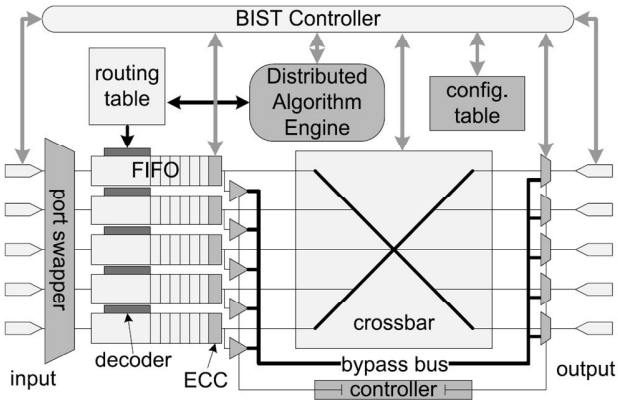

Fig. 6. The structure of router with fault detection and mitigation mechanisms

The Fig. 7 shows an example, when each of the two neighboring routers has one failed port. In one of routers, the input port is failed, in other – the output port. The port swapper allows to loose only one connection in this case (without the port swapper the system will lose two connections). However, use of a port swapper can have a negative effect because it leads to a tangible increase of interconnection lines.

In [5] a NoC with support QoS and fault mitigation is proposed. The virtual channels for data flows with different priorities are supported in this NoC. When a fault occurs, the resources basically used to QoS support are redirected to allow data transmission without QoS. The basic variant of architecture supports two virtual channels. The structure of router is represented in the Fig. 8.

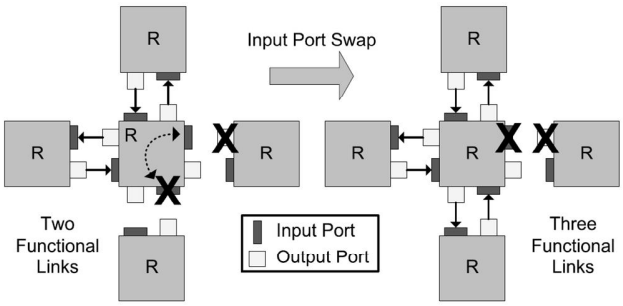

Fig. 7. The example of port swapping

The router has five groups of ports: northern, southern, eastern, western, and local. Every group includes two ports (channels). The first channel is used for high priority traffic; the second channel is used for low priority traffic.

For each group of ports the multiplexing block is implemented. This block allows connecting the external interconnection lines to different FIFO – this function looks like function of port swapper from previous example.

If the group of ports is completely fails, addressed to this direction packets are forwarded to other directions due to routing algorithms.

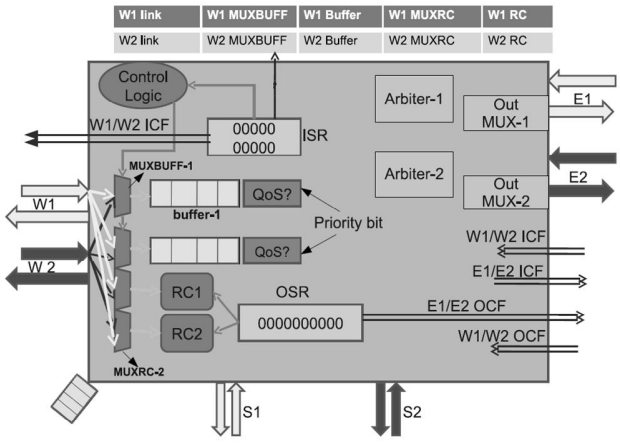

Fig. 8. The structure of router with support QoS and fault mitigation

In [11] authors discuss the various implementations of a router with spare connections. The possible variants for 2D grid are represented on the Fig. 9.

All described above approaches include a fixed set of fault mitigation mechanisms. It is not suitable for NoC, implemented for the applications under consideration. It may lead to unnecessary overhead for implementation of mechanisms that will not be needed, while necessary mechanisms could be absent. For example, if the NoC should be implemented with 180 nm design rule, as shown above, the essential protection of long data wires and buffers are not required. However, if we plan to use 90 nm for the NoC implementation we need strong protection from faults in these components.

In the following sections we describe proposed method that allows to eliminate these shortcomings.

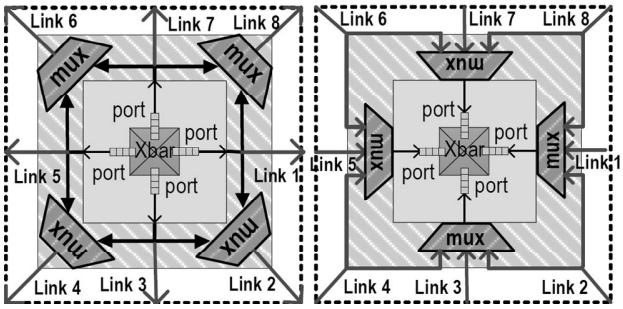

Fig. 9. The possible variants of a router with spare connections for 2D grid

V. THE PROPOSED METHOD OF RECONFIGURABLE NOC STRUCTURE ORGANIZATION WITH FAULT MITIGATION

In [12] we proposed an approach to development of reconfigurable NoC with QoS support but without fault mitigation support. The NoC structure has a reconfiguration of interconnection graph (the regular and irregular structures are supported), has a number of virtual channels in accordance with the characteristics of data streams for running applications.

This system has been focused on the design rules, starting with 180 – 120 nm. NoCs with 10 – 20 terminal nodes can be built on this technology. The basic version of the structure supports strong technology limitations on the permissible length of wires, number of vias. Therefore, it can be used for systems which are manufactured in russian fabrics. An extended versions can be used for embedded systems with large number nodes, implemented with thin design rules. These features are ensured by the parameterization of basic NoC components. The quantity of ports and interconnections, the buffer's size and configuration in RTL model of NoC are set by parameters. The permissible values of these parameters are determined by the capabilities of technology.

Our NoC structure includes tree main types of components: terminal nodes, routers and channel switches. The terminal nodes are connected only to routers. The routers can directly connect to other routers or to channel switches (set by configuration).

The general interconnection structure between the routers and channel switches is represented on Fig. 10. In this figure, diamonds correspond to routing switches, octagons correspond to channel switches. The Fig. 11 includes structure of a fragment of the NoC with one terminal node and the internal structure of routers and channel switches. A set of two or three interconnections is connected to every port of router or channel switch via the multiplexor. At one point of time active only one of them will be. A current configuration of the NoC is determined by the list of active interconnections and by configuration of channel switches. The routers are connected as 2D-grid by direct interconnections. Other graphs can be built by using channel switches.

This structure may be dynamically reconfigurable. Different parts of it can be configured independently. The interconnection graph that is formed as a result of configuration may have regular structure (e.g. torus, binary tree), irregular structure or hybrid structure with regular and irregular zones [12].

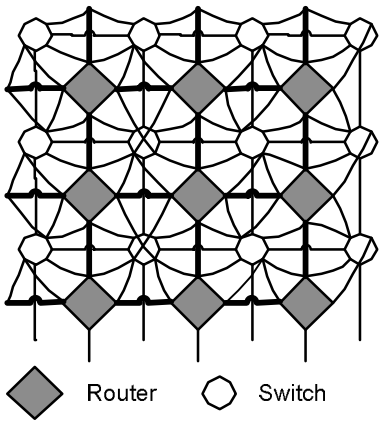

Fig. 10. An example of NoC structure (interconnection between routers and channel switches)

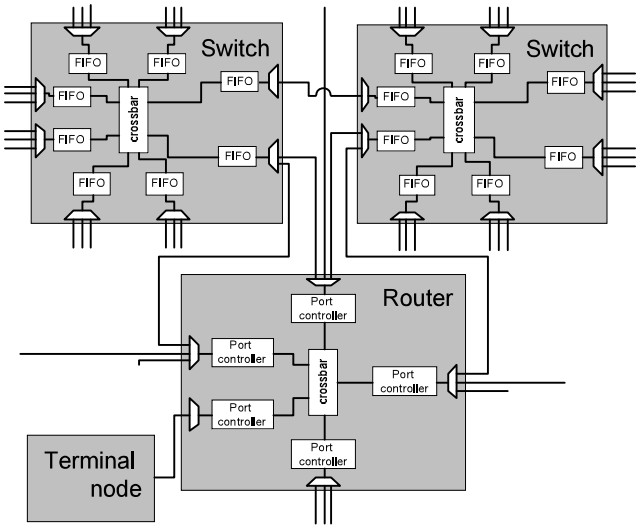

Fig. 11. An example of interconnection between router, channel switches and terminal node

Quantity of interconnection graphs that can be constructed on the basis of the proposed reconfigurable structure depends on the number of ports in channel switches and routers, the number of communication lines. The basic variant represented on the figure can be configured as two-dimensional grid, torus based on the two-dimensional grid, three-dimensional grid, torus, based on three-dimensional grid, banyan, tree.

Our approach to NoC's structure organization has a number of advantages compared with others approaches [13, 14, 15, 16], where topological switches (they are the functional analogs of channel switches in our version) located directly in ports (immediately before ports) of routing switches. In our approach the channel switches are separate components. They can be placed anywhere in the interconnection graph (and the floorplan) independently on placement of routers. It allows:

- To reduce geometric length of interconnection lines. The channel switch includes crossbar that consists of logic gates and can include FIFO. The logic gates act as signal amplifiers (repeaters). Therefore the geometrical length of interconnection lines is reduced in comparison with the variant when switches are placed in ports of routers. Furthermore, the FIFO in channel switches allows to adjust the achievable clock period duration.
- To control the size of crossbar – in case of strong technological constraints we can use a group of channel switches with small number of ports instead of one channel switch with big number of ports.
- To control the interconnection multiplexer size in the routers and channel switches.

These features are very important when the developed NoC should meet strong technology constraints on the interconnection lines length, on the clock period, on the size of the multiplexing blocks and the crossbars.

In this paper we propose a method for development of NoC with faults mitigation based on the discussed above approach to the design of NoC's structure, which is complemented by a series of important features.

The proposed method is focused on different technologies, design rules, within in considered set of application fields, therefore a variety of protection mechanisms can be used for the NoC's components (Table II).Implementation of this variety is archived using the parameterization of RTL model.

A. Fault mitigation for the NoC's components (interconnection lines, channel switches, routers)

For a given technology and operation conditions the probability of failure in long connections between the terminal nodes, the channel switches and the routers depends on their length and bit width. Necessity of faults mitigation for a particular technology can be determined from Table II. As can be seen from it, different technologies require different protection mechanisms.

The main mechanism selected by us is based on parity bits and parity checking. This mechanism allows accurately error localization, it can be implemented in components that do not include buffers (if a buffer is shut down because of errors) and implementation overheads are negligible small (less than 1%).

However, this mechanism does not provide protection incase of crosstalk. For fault mitigation in this case we may use Hamming or others special codes that are applied to individual flits. These codes, as well as parity bits mechanism, do not require buffering. Additional lines and coding/checking blocks for these codes are placed in the NoC structure in same positions as for parity mechanism. The support of all these mechanisms is provided by parameterization of RTL model.

Majority circuits for control signals, a set of spare data lines (and/or mechanisms that allow data transmission via small number of data lines in case of failures) may be included in interconnections also. Enabling/disabling of these mechanisms is provided by parameterization of RTL model. The control circuits for these mechanisms are placed in the ports of routers

and channel switches. It minimizes implementation overheads of (the overheads are less than 3% for a coefficient of duplication 3).

Buffers are used in routers and channel switches. The various mechanisms (see Table 2) may be used for fault mitigation in these blocks. Since in our NoC the switching "on the fly" can be used and buffers may be shut down due errors the CRC based approach cannot be used. In accordance with selected technology and operating conditions the parity bits based approach or approaches, based on Hamming or other special codes can be used. For buffers can used other mechanisms than for interconnection lines.

The channel switch. The channel switch includes interconnection multiplexers (combinational circuits) and buffers for every port, and crossbar (combinational circuit). For all considered technologies fault mitigation in buffers is required. Very small buffers (2 - 4 flits) are used in the channel switches. Therefore they are shut down as a unit when one of its memory cells fails. Bypass of the crossbar not implemented, because the probability of failures is very low: crossbar includes only logic gates that are most resistant to failures, the number of ports is small, thus area is small. Fault mitigation for combinational circuits is required only for most thin of design rules.

If the crossbar fails, the data will be sent via others channel routers. Our approach is focused on using the channel switches with small number of ports. These channel switches occupy very small area. They can be grouped in NoC structure; spare channel switches may be included in the group.

This approach allows to essentially reduce the load of the rest of the system in case of failure in one channel switch, and to reduce the area occupied by spare channel switches. For example in the basic structure with 8-input/output channel switch, it may be replaced by a group of four channel switches (Fig. 12).

The area and timing parameters of both variants are practically equal. Quantity and size of FIFO blocks in the group is same as in 8-th ports switch, summary crossbar's area in the group is same as in 8-th ports switch. The group has additional multiplexors, but its area is negligibly small. The data goes via additional interconnections and additional crossbar in the group of switches. But delays in interconnections are negligible small due they are very short, and summary delays in two 4-th ports crossbars are equal to delays in 8-th ports crossbar. Due the multiplexors in every port, the group of switches translate data as non-blocking switch. However, the probability of an error in each of the channel switches in a group is 4 times less than in the 8-ports channel switch. If one channel switch from group fails, the others can continue functioning correctly. This can significantly reduce number of redirected data streams in case of failure in a channel switch. Therefore the overload of the rest channel switches is decreased. If we include in the group one spare channel switch it's area increases in 1,25 times. If we use one 8-th ports channel switch and include one spare channel switch, the area increases in 2 times. Thus, the gain in area from the use of our approach in this example is 1,6 times.

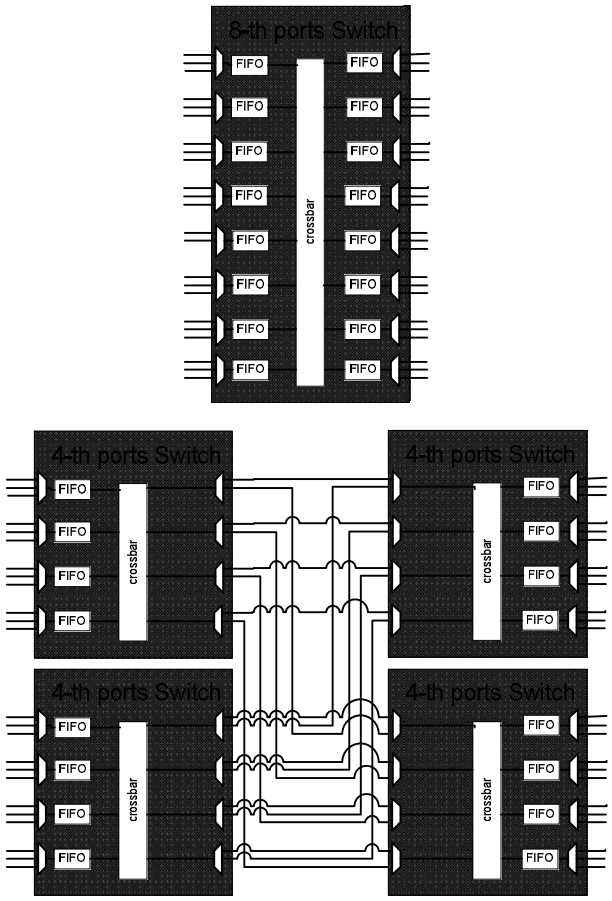

Fig. 12. An example of replacement of one 8-th ports channel switch to group of four 4-th ports channel switches

Router. A router includes the multiplexer of interconnections (combinational circuit) and buffers for every port, the port controllers that rout the packets (FSM) and the crossbar (combinational circuit).For buffers partial shutdown mechanism is provided in case of individual cell's failure. In the routers could be used the significantly larger buffers than in the channel switches.

Grouping and including a spare router in a group approach also can be used for routers. But the routers area is essentially bigger than the channel switch area; therefore this approach is not the main one. For systems in which the overhead of this approach implementation is unacceptable, we use including of a row of spare routers. However, in contrast to the approach proposed in [7], we suggest to add not horizontal but diagonal rows. For used interconnection structure this approach allows to reduce number of reconfigurable routers in case of one router failure in two times (by reducing the average distance between the failed router and the spare router). The Fig. 13 shows an example of placing the spare routers (highlighted by hatching). A terminal node should be connected to several routers.

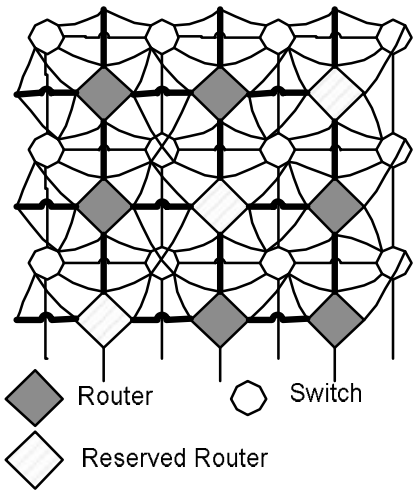

Fig. 13. An example of placing the spare routers

B. *The method of reconfigurable Noc structure organization*

1) In accordance with the selected technology, the operating conditions and taking into account the areas of blocks to determine the fault mitigation mechanisms for interconnection lines.
2) In accordance with the selected technology and the operating conditions to determine fault mitigation mechanisms for buffers.
3) In accordance with user application set, required logical channels graph and through put of logical channels to determine the NoC structure basic variant.
4) In accordance with the selected technology, the operating conditions and taking into account the areas of blocks to determine allowable number of ports in channel switches, to determine the need for redundant channel switches. If the channel switches in the basic structure have many ports, then to replace every such channel switch to a group of channel switches with allowable number of ports. To include redundant channel switches in groups if need.
5) In accordance with the selected technology and the operating conditions to determine necessary quantity of redundant routers, to include these routers in NoC.
6) In accordance with the selected technology and the operating conditions to determine number of routers, to which should be connected to every terminal node. A terminal node is connected to the nearest routers.

VI. VERIFICATION OF FAULTS MITIGATION IN THE FRAME OF PROPOSED METHOD

Verification of the proposed method of reconfigurable NoC structure organization was carried out with using analysis and simulation.

A. *Analisys*

The process of formal verification is a very time consuming, so in most cases it is carried out for the individual component, not for the system as a whole. Formal verification was carried out for models of components with single error and for models of interconnections with double errors. Possibility of multiple errors in combinational circuits and in memory cells is very low for the considered technologies.

During the analysis it the list of possible errors (based on Table I and Table II) and scenarios of its processing were defined. It has been shown that the errors in all components, for which fault mitigation was suggested, do not lead to errors in NoC functionality or are identified and localized (don't lead to deadlocks, or errors during others data packets transmission).

B. *Verification using the simulation models*

Currently there are a large number of CAD tools for simulation of effects of failures on the behavior of the SoC, NoC. The most widely used tools are Synopsys Tetra MAX ATPG, Mentor Graphics Tessent Fast Scan, Mentor Graphics Tessent Test Kompress, Cadence Verifault, Cadence Encounter True-Time ATPG, Syntest Turbo Fault, Winterlogic Z01X. These tools generally support a similar set of possible faults and same fault injection schemes.

For example, let's consider Cadence VeriFault [17]. This tool operates with the logical or physical netlist. It supports faults injection such as Stuck-At 0 (constant "0") and Stuck-At 1, broken lanes and bridges between lanes.

These tools allow to select types of errors and the area of the project (SoC) in which they may occur. Before simulation the tool creates a copy of the original model and injects the selected faults in the selected regions. Then simulation of the original and the modified models is made with the same tests. After that the tools evaluate the percentage of failure detection and the list of faults that are unobservable.

However, these tools don't allow simulation of dynamically appeared faults. This is due to the fact that these tools, like most currently available CAD, are focused primary on simulation of the manufacturing defects.

There is a number of approaches for elimination of this shortcoming. In the most of them special blocks are inserted in the model. These blocks distort a state of memory elements during the simulation [18]. The ways of distortions may be different. Two approaches are most widely used. In the first approach the special components writes the distorted values to memory elements via the main communication system of the NoC model.

In the second approach the special components uses the special functions of EDA tools to make distortions. For example, Cadence EDA includes nc_force functions for this goal. Also Cadence EDA supports direct references to the elements (in particular, modifications of values) of the model by their hierarchical names (only for models, written on Verilog).

These approaches have two main disadvantages:

- the need to incorporate into the model (RTL or netlist) the specialized components, which could lead to bringing functional errors into this model;
- errors can be embedded only in memory elements, and can't be embed in combination logics and interconnections.

We suggest a new approach to faults simulation that allows overcoming these limitations. In every library component we embed the structures:

- the counter 1;
- the counter 2;
- the conditional operator for fault generation.

The rule of count is set via parameters. The counter 1 is used for:

- count of change values on the inputs of the component;
- count of the active clock edges (for the flip-flops);
- count simulation time with specified precision (ps or ns).

The counter 2 is used for count of the duration of fault's time with specified precision (ps or ns). The rule of count is set via parameters. The option to set value corresponding to infinity provides possibility of hard faults simulation.

The conditional operator for fault generation provides inversion of the output value (for Boolean output the incorrect value = the inverse value). The condition of this operator is determined as function of the counter 1. The model can include some functions of failure time, for example, the uniform distribution, exponential distribution and others. The used variant of the function and its parameters are specified parametrically. The failure duration time is determined by the counter 2.

The set of library components can be divided in some groups. User can define the types of possible faults and its parameters for every group. This approach allows in particular simulation of glitches on outputs of the components. It can be used also for simulation of glitches in communication lines, connected to these outputs.

The user needs to embed the fault models in the library components' models for each library only once, not in every SoC project. Parameters of faults can be placed in a separate packet. Its values can be set for every concrete project correspondingly to its operation conditions.

To prove the suggested approach on the RTL models were taken that include from 4 to 16 routers and from 16 to 64 channel switches. The terminal nodes are represented by data packets flow generators and controllers. The data packet flows parameters are set via parameters.

We synthesize these models by Cadence RTL Compiler. Then we simulate the resulting netlists with Cadence Verifault and the component library with our modifications for dynamic faults simulation. Simulation time with the modified library is only 10% more than for the basic variant of the library. Thus, the proposed approach to the simulation of dynamic emerging faults has acceptable performance.

We simulate the models with loads of the NoC from 40% to 90%, the soft faults possibility from 10^{-12} to 10^{-6}, the hard faults possibility from 10^{-15} to 10^{-12}. These ranges of the possibilities are selected on the ground of the data, represented in [1]. The deadlocks and the livelocks of packets don't appear during the simulations. The number of lost and corrupted packets don't exceed 3% from the total quantity of the packets for the soft faults probability = 10^{-6}, the hard faults probability = 10^{-12}, and don't exceed 1% for the soft faults probability = 10^{-12}, the hard faults probability = 10^{-15}.

In frame of proposed approach only a modification of the library components source code is required with using of standard Verilog language. Therefore it can be used not only in combination with Cadence EDA but also with others EDA, for example, from Synopsys.

VII. Conclusion

In this paper we consider possible sources of faults in NoC for embedded systems manufactured using thin design rules, describe the existing methods for faults mitigation.

We suggest an approach for development of the NoC with dynamic reconfigurable structure for the fault mitigation. Unlike existing approaches, it is focused on the use of various technologies under consideration by supporting various fault mitigation techniques.

We prove this approach with using analysis and simulation. We suggest an approach to simulation that allows to cover dynamic emerging errors in a NoC model. We apply this approach in Cadence Verifault tool that in its basic variant allows simulation of only static errors.

Our directions for further work are:

- development of the proposed method for 3D NoC;
- development of NoC dynamic reconfiguration algorithms that would be oriented to minimize overheads, especially the amount of information transmitted during the reconfiguration and the reconfiguration time;
- development of the proposed simulation approach – support for MCU simulation.

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

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

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

[4] SMAH Jafri, L. Guang, A. Hemani, K. Paul, J. Plosila and H. Tenhunen, "Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes", *Microprocessors and Microsystems*, vol. 37, issue 8, 2013, pp. 811–822.

[5] M.R. Kakoee, V. Bertacco and L. Benini, "ReliNoC: a reliable network for priority-based on-chip communication", *in Proceedings of the design, automation and test in Europe conference (DATE)*, 2011, pp. 1–6.

[6] M. Koibuchi, M. Hiroki, A. Hideharu and T.M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip", *in Proceedings of the international symposium on networks-on-chip (NOCS)*, 2008, pp. 13–22.

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

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

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

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

[11] Animesh Jain, Ritesh Parikh and Valeria Bertacco, "High Radix On-Chip Networks at Incremental Reconfiguration Costs", *23rd International Workshop on Logic and Synthesis*, 2014.

[12] E.A. Suvorova, Yu.E. Sheinin and N.A. Matveeva, "Methods of design the reconfigurable communication systems for networks on crystal, developed on 3D technology", *Scientific Journal of "Proceedings of the Samara Scientific Center of the Russian Academy of Sciences"*, vol. 16, issue 6(2), 2014, pp. 605–611.

[13] M.B. Stensgaard and J. Sparso, "ReNoC: a Network-on-Chip Architecture with Reconfigurable Topology", *in Proceedings of Second ACM/IEEE International Symposium on Networks-on-Chip*, April 2008, pp. 55–64.

[14] Radu Marculescu, Jingcao Hu and Umit Y. Ogras, "Key Research Problems in NoC Design: a Holistic Perspective", *in Proceedings of third IEEE/ACM/IFIP International Conference Hardware/Software Codesign and System Synthesis*, Sept. 2005, pp. 69–74.

[15] Mehdi Modarressi and Hamid Sarbazi-Azad, "Power-Aware Mapping for Reconfigurable NoC Architectures", *in Proceedings of the 25th International Computer Design Conference*, Oct. 2007, pp. 417–422.

[16] U.Y. Ogras and R. Marculescu, "Application-Specific Network-on-Chip Architecture Customization via Long-Range Link Insertion", *in Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2005, pp. 246–253.

[17] Verifault-XL® User Guide, Cadence Inc, 2014, p. 220.

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