Design and Simulation of Onboard SpaceWire Networks

Valentin Olenev, Irina Lavrovskaya, Ilya Korobkov, Yuriy Sheynin
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
{valentin.olenev, irina.lavrovskaya, ilya.korobkov}@guap.ru, sheynin@aanet.ru

***Abstract*—The paper describes SpaceWire Automated Network Design and Simulation (SANDS) – the new CAD system for SpaceWire networks. SANDS system supports the full on-board network design and simulation flow, which begins from the network topology automated generation and finishes with getting the network structure, configuration and parameters setting, simulation results and statistics. The paper also provides use cases for SANDS application.**

I. INTRODUCTION

Nowadays, onboard networks are composed from a big number of elements with various functionality. All these elements are interconnected with each other via onboard network communication infrastructure. Onboard networks are now used in spacecrafts and in avionics. A common technology for this kind of networking is SpaceWire , which is easy to understand, it is compact and efficient in implementation [1].

It is easy to build and operate SpaceWire networks while they are small enough, while they are used for well predictable and not complicated set of information streams without strong requirements and constraints. However, as a network becomes larger, over some dozens of nodes, information streams becomes more dense, requirements and constraints stronger to design and operate SpaceWire network becomes tough tasks.

The motivation for the conducted project was a need to create a complex of software tools for computer-aided design and modeling for SpaceWire networks. This tool suite is aimed to:

- Reduce the time-to-market;
- Decrease the cost of design and verification of hardware;
- Improve the design quality as a result of detailed and deep elaboration of technical solutions and simulation opportunities.

The paper describes SpaceWire Automated Network Design and Simulation (SANDS) – the new CAD system for SpaceWire networks. In section II we introduce SANDS and its components. Section III is referred to an overview of related studies. In next sections we present all components in details with demonstration use cases.

II. SPACEWIRE AUTOMATED NETWORK DESIGN AND SIMULATION

SANDS is a computer-aided design system for SpaceWire networks [2], which supports full network design and simulation flow, which begins from the network topology automated generation and finishes with getting simulation results and statistics. SANDS includes almost of the functionality that is needed for the prototyping a real on-board network. SANDS architecture is shown in Fig. 1.

SANDS architecture includes four main components:

- Component #1: A component for onboard network topology design and evaluation of its structural characteristics;
- Component #2: A component for tracking of the non-intersecting routes for the data transmission in a network;

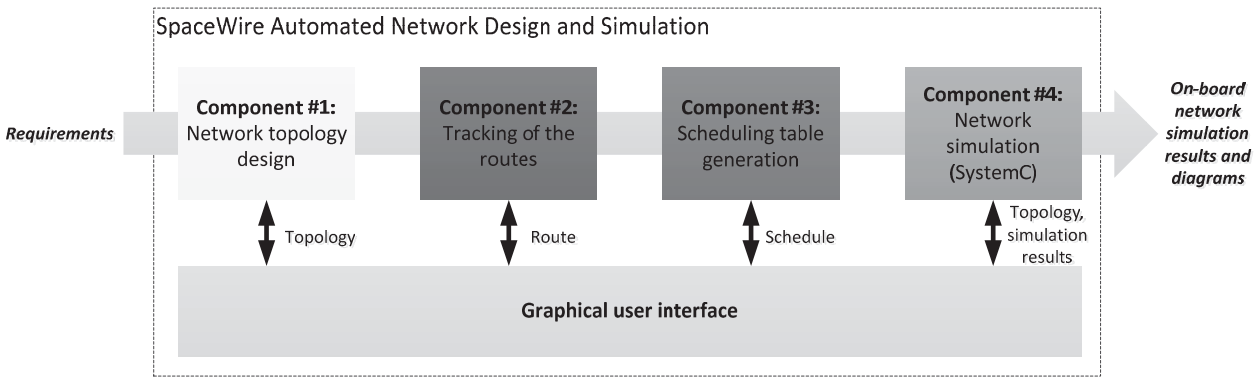

Fig. 1. Architecture of SpaceWire Automated Network Design and Simulation toolset

- Component #3: A component for generation of the scheduling table for the STP-ISS transport protocol for the transmission of the data with Scheduled quality of service;
- Component #4: A component for simulation of the network operation with all the data that component got from other 3 components and graphical user interface.

We implemented a flexible system. It is adapted to have a possibility to add new protocols. Addition new protocols requires some efforts. Moreover, there are some protocol restrictions. For example: if you want to add new transport layer protocol, it should be able to operate over already implemented network layer protocol. Currently, we included implementation of the SpaceWire protocol and two transport layer protocols: RMAP [3] and STP-ISS [4]. Now SpaceWire-oriented transport protocols like PTP [5], JRDDP[6], STUP[7], STP [8] could be added without significant modifications of system, because SpaceWire network layer was already included.

Visualization and user interface is provided by a graphical user interface (GUI), developed in scope of the VIPE project [9]. This interface provides the visual network composition and management capabilities. It allows designing SpaceWire network topology in visual interactive way from such elements as terminal nodes, routers and links (see Fig. 2).

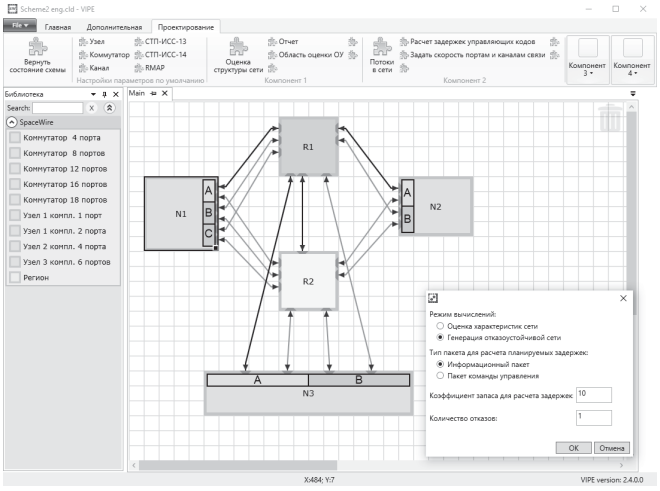

Fig. 2. SANDS graphical user interface

The library of elements is a replenish set of network nodes and switches relevant to physical devices that are available for network building. It may also include flexible components if the developer wants to try various combinations of network equipment. For all nodes, switches and channels the GUI will provide the configuration interface to set up their parameters, configure transport protocols and application-level traffic generators.

The designed network will be exported to the intermediate representation format to be used in other SANDS components for simulator, routes tracking, scheduling calculation and other tools. GUI will be also able to show results after running each component.

III. Related work

Let us consider related studies for each component of SANDS system.

A. Fault-tolerance and network topology reconfiguration

Among a wide range of modern CAD systems for network and SoC design there are no such CAD which provide facilities for fault tolerance analysis of the designed network [10].

In [11] it is stated that a fault tolerance analysis feature is provided by Fault Tolerance Simulation and Evaluation Tool for Artificial Neural Networks (FTSET) which is based on Matlab. This tool is capable of evaluation the fault tolerance degree of a previously trained Artificial Neural Network given its inputs ranges, the weights and the architecture. However, this FTSET evaluates graceful degradation characteristic but not connectivity characteristic of a network, which is a subject of interest of this paper.

There is no other information in the open sources, than [12], that considers fault tolerance for SpaceWire networks. In that paper, authors propose the methodology and a toolset for SpaceWire network design. This methodology allows generating fault tolerant networks with variable level of tolerance. However, there are no fault-tolerance analysis algorithm provided in this paper.

B. Deadlock-free routing

In paper [13] authors propose a solution for deadlock-free routing problem. Authors present an algorithm of breaking cycles in CDG. If a deadlock exists, it is removed by virtual channel insert. The presented algorithm does not restrict a designer in choosing the routing, but it requires some resources and technologies that cannot be applied in some networks. For example, it is impossible to solve deadlock-free routing problem by virtual channel insert for SpaceWire network, since this standard does not support virtual channel technology. Such strategy can be used only if deadlocks are rare and it is possible to remove them by introducing additional resources [14].

C. Scheduling tables creation

There are two software tools, which provide functionality close to the one from SANDS Component #3. They are MASIW tool suite and TTEthernet scheduling tool.

Let us consider MASIW first [15]. This tool allows to generate the distribution of the processor's computing time between functional applications. As a result it produces a schedule of launching applications for ARINC-653 compatible real-time operating systems. It should be noted that MASIW produces a schedule for performing strictly periodic tasks, i.e. such a schedule in which the time between adjacent runs of one periodic task is fixed and equal to the length of the period. The tasks solved in MASIW, and the problems solved in Component #3 in SANDS, have some similarities. The main similarity is the periodicity property: information flows in onboard network are characterized by periodicity, as well as the tasks of launching applications on the processor. However, the main objective of the software tools is different: while MASIW

forms timing diagrams of the application launch schedule on the processor, the SANDS scheduling component generates a schedule for data transmission by the end nodes in the onboard network.

Another software tool for scheduling is TTEPlan from TTTech company [16]. TTEPlan – is a commercial tool for scheduling in TTEthernet / AFDX [17], [18] networks. TTEPlan solves similar problems with those that are solved in scheduling component in SANDS. However, TTEPlan solves these problems for networks based on TTEthernet and AFDX standards and does not allow operate with SpaceWire networks. Moreover, TTEPlan is a commercial product that cannot be introduced and used within the a computer-aided design system for SpaceWire onboard computer networks.

D. Network simulation

Network simulators allow researchers to test the scenarios that are difficult or expensive to imitate in real world. It is particularly useful to test new communication protocols or to change the existing protocols in a controlled and reproducible environment. Simulators can be used to design different network topologies using various types of nodes.

Let us consider some tools that are specially developed for SpaceWire networks modeling. Most of them are based on the OPNET [19] simulation framework. For these purposes OPNET was adapted for the SpaceWire and updated with a list of specific modules and network elements. This way was chosen by Thales Alenia company, which implemented MOST (Modeling of SpaceWire Traffic) [20], [17] for the European Space Agency. Initially MOST was based on the OPNET toolkit dedicated to network modeling. Currently, MOST is available for NS3 simulator [22].

Similar solution has been developed by Sandia National Laboratories (SNL) [23], but it gives less abilities than MOST, because it does not have an option to insert errors to the transmitted data, which is not good for testing and verification. The SNL team also used extension features within OPNET Modeler to create a set of general purpose modules representing different network elements or basic building blocks for SpaceWire networks simulation. The second ability of the tool is an in-depth analysis of the accurate distribution of system time across the SpaceWire network.

However, there is a tool that is not based on the OPNET. It is VisualSim SpaceWire modeling, developed by MIRABILIS Design Company [24]. VisualSim is intended for end-to-end system-level design. VisualSim gives an ability to test the real hardware SpaceWire devices, but it is not applicable for prototyping of real onboard networks on early stages of the project.

Consequently, there are only three tools that give an ability to build SpaceWire networks and to simulate their operation. Two of them are based on the OPNET and use its capabilities. Moreover, all these three tools are commercial and rather expensive.

IV. Network topology design and evaluation of its structural characteristics

A. General description

SANDS provides strong network structure analysis features. Component #1 is responsible for the following tasks:

- network topology design;
- evaluation of structural characteristics of the designed topology;
- network topology transformation for achieving required fault-tolerance.

Network topology design assumes creation of a network topology by means of GUI and setting up parameters for nodes, switches and links. Once this step is completed, the designed network can be analysed. This component analyses the following characteristics of the designed network:

- Network mass: cable mass, switches mass or both;
- Power consumption of network switches;
- Network diameter
- Fault-tolerance of the network or its cluster.

While estimation of the first three characteristics is rather easy, the last one is worth additional research and discussion. Fault-tolerance is a very important characteristic for onboard networks, especially for the domain of long-term satellites. It is a common practice when the network topology contains redundant nodes and links.

Nowadays, sizes of the network topologies increase with the growing demands of industry. That is why it is so important to obtain an automated estimation of fault-tolerance characteristic in the designed network and remove the human factor for the most part.

In our computer-aided design system the fault tolerance analysis of onboard networks is evaluated on the basis of vertex connectivity of the graph [25]. If the value of fault tolerance is 0 then SANDS will search all bottlenecks of the designed network.

Another key feature of the Component #1 is network topology transformation for achieving required fault-tolerance. The problem can be described as follows. Given a network topology, place additional routers and communication links to achieve the required by user fault tolerance. Such a transformed topology is obtained in two stages [26]:

- Stage 1: Initialise a topology of a network with required fault tolerance f.
- Stage 2: Iterative improvement of the topology obtained on the stage 1.

Network transformation can be done either for the whole network topology or for its part. Moreover, we specified a solution for a network consisting of several SpaceWire network regions.

B. Use cases

Firstly, we provide a use case of analysis features of SANDS. Fig. 3 shows a network which consist of 12 terminal nodes and 6 routers.

After performing analysis SANDS gives a detailed analysis report including mass parameters for routers and cables, power consumption estimation, minimal and maximal link rates. Moreover, it performs fault-tolerance analysis which is one of the key features of this analysis [25].

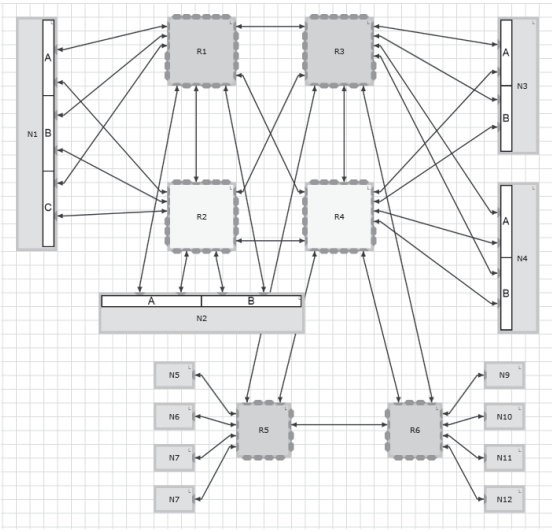

Fig. 3. Example of network topology analysis

Fault-tolerance analysis for this network structure results in zero fault tolerance. There is a bottleneck in this network in its bottom part containing the following nodes and routers: N5 – N12, R5 and R6. This part of a network cannot tolerate any fault. The user is provided with all this information in the report and corresponding highlights in GUI.

Next use case is referred to the network topology transformation feature. Let us show an example of algorithm application to a network with regions (see Fig. 4). The initial network topology consists of three regions. The whole network is not fault-tolerant, however, Region 3 is *1*-fault-tolerant, because of cross-connections between nodes' units and routers R3 and R4.

In the discussed topology it can be observed that nodes N2, N3 and N4 have two ports, but only one port is connected to the router.

We ran SANDS software to obtain a network which is *1*-fault-tolerant. The result of network transformation is shown in Fig. 5. All new links are shown in heavy lines and new routers are shown with dark frames.

Our network transformation algorithm connected free ports of terminal nodes N1, N2, N3 and N4 for cross-connections. Each terminal node in Region 1 and Region 2 contains two SpaceWire ports which are connected to different routers in order to increase fault tolerance.

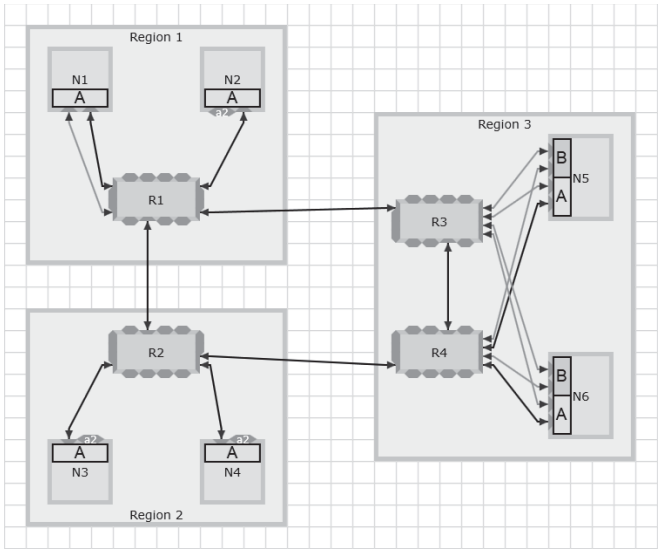

Fig. 4. Example of network transformation

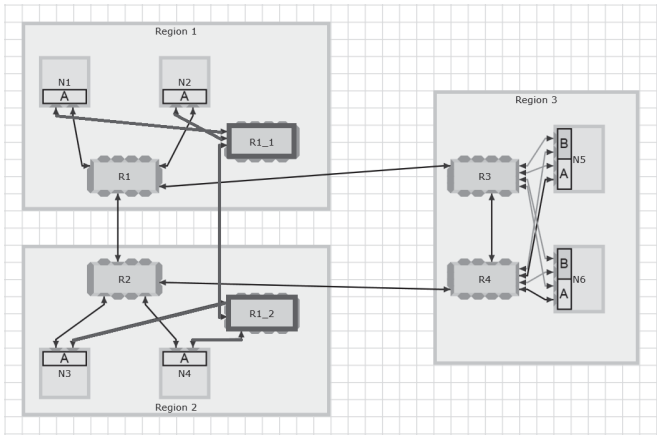

Fig. 5. Transformed topology

Moreover, we can observe two new routers R1_1 and R2_1, which were added to increase fault tolerance in Region 1 and Region 2. Topology transformation tool added 5 additional communication links. Region 3 was not transformed as it was originally *1*-fault-tolerant. The resulting network topology is *1*-fault-tolerant.

V. TRACKING OF DEADLOCK-FREE ROUTES FOR THE DATA TRANSMISSION IN A NETWORK

A. General description

Another SANDS functionality is data transmission routing in a network without deadlocks. This is one of the most important tasks in network design; it implies finding sequence of switches between source and target devices and creating routing tables for these switches. SpaceWire technology provides opportunity of creating onboard computing network with flexible network architecture and high throughput. Adjusting routing tables and adaptive routing tables in switches makes possible of using static routing in SpaceWire network.

The deadlock-free routing in static routing provide guarantee that packet transfer will be successfully finished in the target device, if following conditions are observed:

- There is the path in the network between source and target nodes, that can be used for data transfer;
- All physical channels and switches are stable and work correct.

The onboard network architecture changes from mission to mission according to actual tasks. Therefore, designers cannot use solutions that were achieved in previous projects. It means that for each new mission it is necessary to repeat whole process of network design including deadlock-free routing and creating routing table. This is very difficult and high consumption task.

The objective of the second SANDS component is to find the specified by network designer number of routes, which do not lead to deadlocks and satisfy to specified criteria (the maximum data transmission delay and distance between connected devices). It is advisable to solve this task consistently, separating it into subtasks [27]:

- Find all possible routes that do not lead to deadlocks;
- Choose routes that satisfy to criteria.
- We have to note the additional requirements for data transmission routing:
- At least one of routes must use only main network elements;
- Route that uses reserve elements must also use main elements.

In accordance with tracked routes SANDS provides routing tables for configuration of routers in onboard network. In addition, this component of SANDS provides estimation of worst-case data and control codes transmission latencies.

B. Use case

The following use case shows an example of tracking deadlock free routes in a network. Fig. 6 shows a network which consist of 5 terminal nodes and 6 routers.

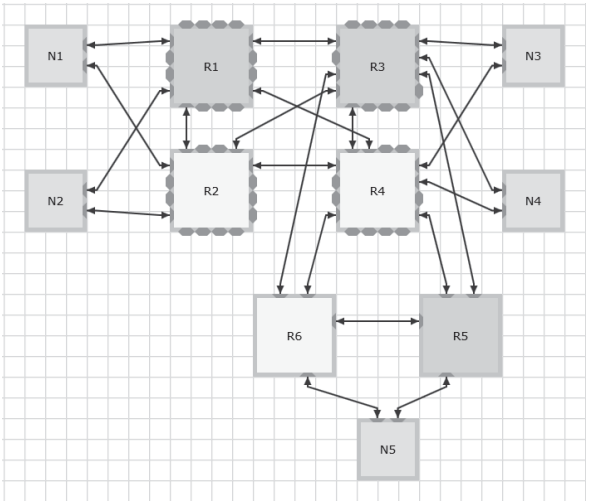

Fig. 6. Topology for tracking of routes

Let us assume the following communications in the network:

- Node N1 sends data to node N4:
- Node N2 sends data to node N3;
- Node N3 sends data to node N5;
- Node N5 sends data to node N1;

We ran Component#2 software to get routes for data transmission. The resulting routes are given below:

- N1 - R1 - R3 - N4;
- N2 - R2 - R3 - N3;
- N3 - R3 - R5 - N5;
- N5 - R5 - R3 - R1 - N1.

Each route can be graphically highlighted for user's convenience (see Fig. 7).

Dark frames for nodes show initiator and target nodes (N1 – initiator, N4 - target). Heavy lines for links and frames for routers show the transmission path.

Moreover, SANDS fills routing tables in accordance to the generated routes combination.

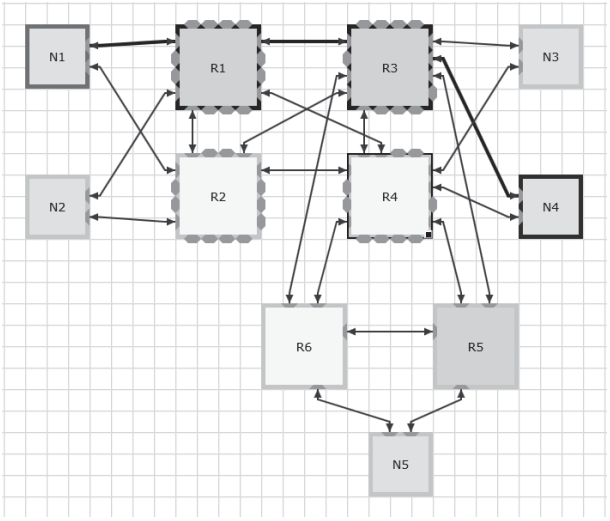

Fig. 7. Route from N1 to N4

In addition, Component #2 calculates data transmission latencies for all defined by user information flows, see Table I.

TABLE I. DATA TRANSMISSION LATENCIES FOR GENERATED ROUTES

Route	Number of hops	Latency, ms
N1->R1->R3->N4	3	0,1666
N2->R1->R3->N3	3	0,1666
N5->R5->R3->R1->N1	4	1,2038
N3->R4->R5->N5	3	0,1413

VI. GENERATION OF SCHEDULING TABLES

A. General description

The problem of scheduling traffic in the onboard network is one of the key issues in the general design flow, since at this stage, using scheduling, you can regulate the information flows and avoid conflicts. Moreover, scheduling allows using the limited resources of the network and also provides guaranteed delays for packet transfer.

Nowadays there are transport layer protocols for SpaceWire that provide time division multiplexing mechanisms (e.g. STP-ISS, SpaceWire-D). Scheduling tables creation is a very complex problem which is very difficult to solve for big networks with multiple data exchanges.

SANDS provides functionality for generation of scheduling tables for STP-ISS scheduling quality of service. These scheduling tables will take into account the network structure and tracked routes.

Component #3 performs the construction of a schedule in which the delivery time of all packets for all traffics in the network would not exceed the maximum allowable latency which was defined by the user. In case this is not possible, the user will receive a corresponding message with recommendations.

Input parameter of Component #3 are:

- Parameters of traffics: route, packets size, period of issue, guaranteed/non-guaranteed delivery (requires ack or not), acceptable latency of packets, acceptable latency of ack-packets; critical/non-critical traffic; order of packets issue;
- Max numbers of time-slots;
- Max duration of 1 epoch;
- Max duration of 1 time-slot;
- Time to switch time-slots;
- Max channels bandwidth usage in percentage;
- Size of extra tail of packets.

This software provides the following functionality:

1) Scheduling of information flows transmission;
2) Generate the scheduling tables for the STP-ISS transport protocol;
3) Configuration should be done taking into account the network and structure tracked routes.

In general, the schedule creation algorithm can be represented in the following steps:

Step 1. Getting and validation of input data. If there is an incorrect input data, go to step 7.
Step 2. Checking the possibility of packet transmission with acceptable latencies. Checking whether the rates of links is sufficient to transmit the packet with an acceptable latency. If not enough, go to step 7.
Step 3. Determining the epoch and time-slots duration.
Step 4. Check whether there is enough network bandwidth to transmit all packets of information flows within one epoch. If not enough, go to step 7.
Step 5. Check the need for traffic scheduling.
Step 6. Scheduling - assignment of traffics to time-slots.
Step 7. Output of the Component #3 operation results.

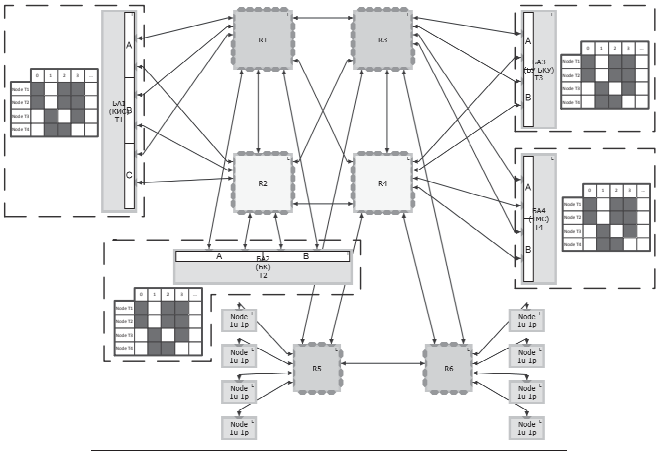

Fig. 8. Schedules for nodes

B. Use case

This use case shows an example generation of scheduling tables. Fig. 9 shows a network which consist of 5 terminal nodes and 2 routers.

Let us assume the following communications in the network:

- Nodes N1 – T4 send data to node N5;
- Node 5 sends data to N1.

Component #3 requires also the following input data to generate schedule:

- Number of time-slots in an epoch. For the current use case we set this number to 14 time-slots.
- Epoch duration. For the current use case we set this duration to 200 ms.

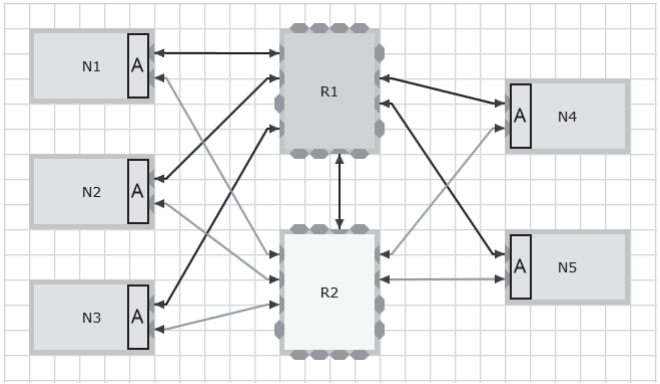

Fig. 9. Topology for scheduling table generation

SANDS Component #3 then successfully generates a schedule for this network, see Fig. 10.

Node ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N1														
N2														
N3														
N4														
N5														

Fig. 10. Schedule generated by SANDS Component #3

As it can be observed from the schedule each traffic will be sent periodically and would not interfere during transmission.

Moreover, the user is provided with a report in with the following information is provided:

- Time-slot duration;
- Time-code relevancy window for STP-ISS-14 scheduling QoS;
- Time-code transmission period.
- Recommendations is case it is impossible to generate a schedule.

VII. SIMULATION OF THE NETWORK OPERATION

A. General description

Finally, SANDS provides simulation features. Simulation core is based on the DCNSimulator extended with additional functionality and is implemented in SystemC. There will be two modes for simulation of network operation with different levels of details (see Fig. 11):

- Bit level – simulation of full SpaceWire stack and Transport protocol or Application.
- Packet level – simulation of upper layers only: Network, Transport, Application.

For bit level simulation SANDS uses SystemC clock based simulation option. This is done in order to achieve a more accurate comparison of the model with a hardware device. Nodes operation in a model closely corresponds to clock signals which impact data transmission latencies in links, ports and switches. However, bit level simulation gives big difference between the real simulation time of the network and time in a model, since the model of each network component describes in detail all the internal mechanisms of the SpaceWire protocol.

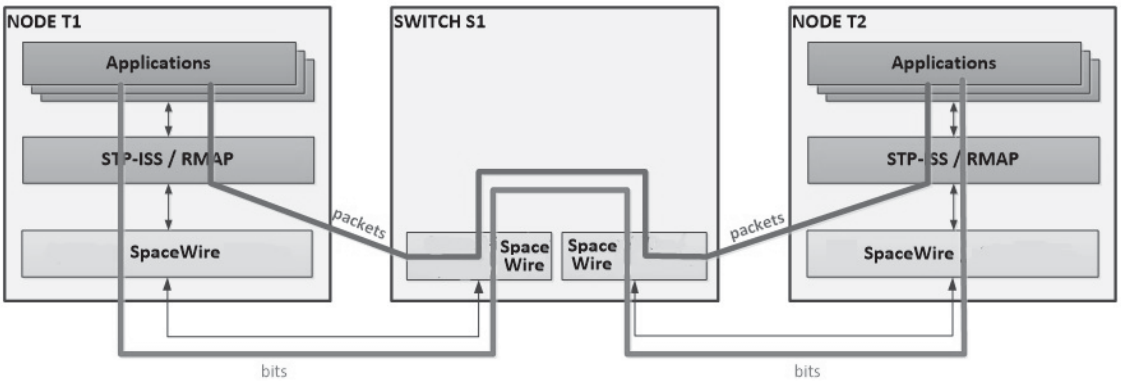

Fig. 11. Two simulation modes in SANDS

Packet level simulation is implemented using event based mode in SystemC. Events, that appear while performing various functions, trigger other functions to be executed. In this simulation mode, the SpaceWire level operation is not considered, which greatly simplifies the model operation logic. Nodes operation in a model depends on various events while all data transmission delays in links, ports and routers are calculated basing on bit level simulation and are defined in correspondent network elements. Such modeling significantly reduces the time for modeling the SpaceWire network, which makes it possible to simulate long periods of onboard network operation.

Finally, the user will be provided with detailed statistical log tracking all events of interest in the network in html format.

B. Use case

This use case overviews simulation of the designed SpaceWire network. This is done by means of the Component #4. This Component takes as an input all the results, that user archived form Components #1-#3. Fig. 12 shows the simple SpaceWire network, that is planned to be simulated. Let us assume that this network has already been designed in Component #1, Component #2 tracked the routes in the network. For such a small network we do not need to schedule the traffic, so we did not use Component #3.

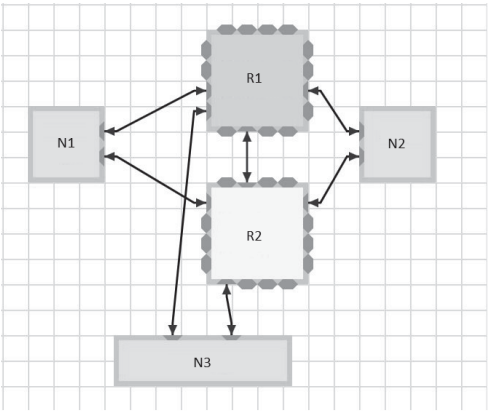

Fig. 12. SpaceWire network topology fot the simulation

To start the simulation firstly the user needs to configure the traffic generators in nodes: add different types of packets to be generated, set the generation periods and different fields of the SpaceWire packets. Then transport protocols in nodes also should be configured. Fig. 13 and Fig. 14 show different forms for setting of packet parameters and timers of STP-ISS transport protocol correspondingly.

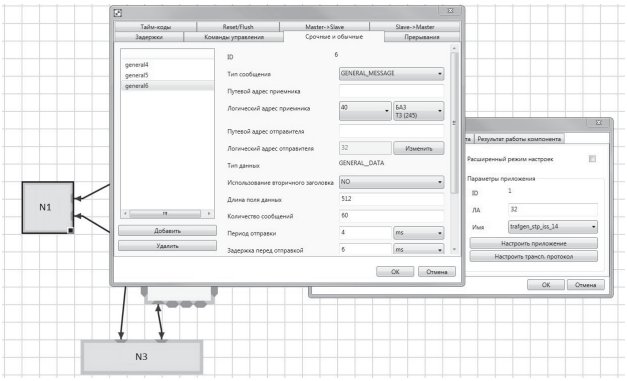

Fig. 13. Setting of traffic generator parameters

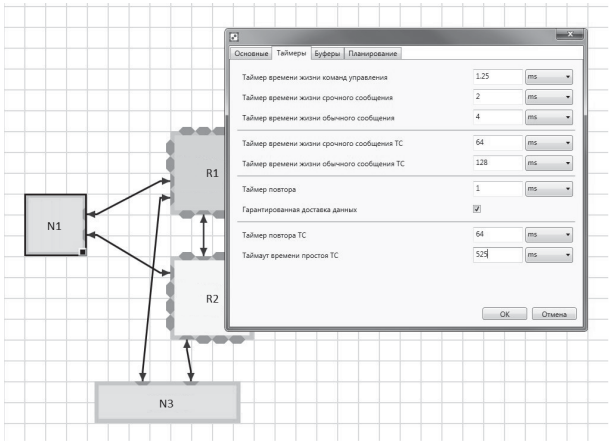

Fig. 14. Setting of STP-ISS protocol parameters

When the network is fully configured at all levels of abstraction (application, transport and data-link), the user can start the simulation. To do this they need to choose the simulation type: bit-level simulation or packet-level simulation, then fill the simulation time field and then start the Component #4 (see Fig. 15).

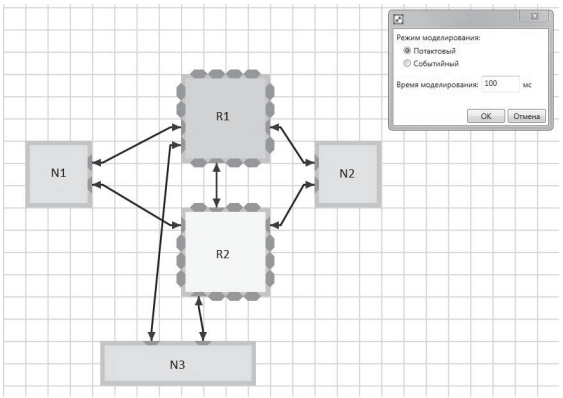

Fig. 15. Starting of simulation

When simulation starts, the system shows a progress bar and the user needs to wait a bit to get the results (see Fig. 16).

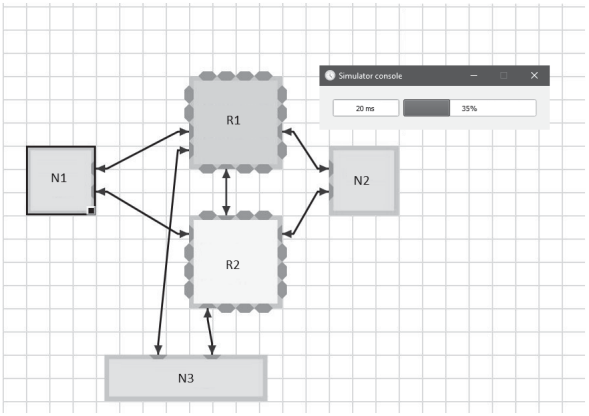

Fig. 16. Simulation process

Simulation results are shown in Fig. 17. Simulation log is represented in HTML form and gives full information for each event in the SpaceWire node, transport protocols, SpaceWire switches, channels and information on errors. All this information is presented in relation with the particular time of an event.

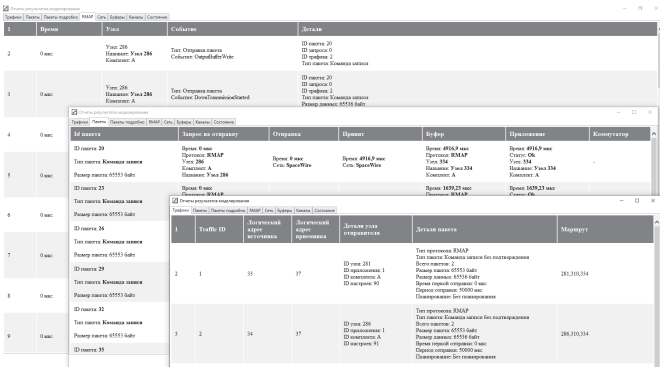

Fig. 17. Simulation results

Current version of the log implementation is not completed yet and finally it would be represented in special SANDS forms with log filtering possibilities.

VIII. Conclusion

In current paper, we present SANDS – a new computer-aided design system for SpaceWire onboard networks design and simulation. This software solves important tasks, which spacecraft developers face with during implementation of satellites and other space vehicles. SANDS begins its workflow from the on-board network topology design and estimation of its physical characteristics. Then it gives an ability to track deadlock-free routes for the data transmission in a network and generate scheduling tables for the STP-ISS transport protocol for data transmission with Scheduled QoS. After the network design is completed – the last stage is simulation of the network operation with real characteristics. Graphical user interface provides possibilities to draw the network topology and set different parameters of nodes, switches and channels. SANDS tool suite is a good solution for the spacecraft design,

implementation and testing. In addition we provided use cases for each component of SANDS.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation within the project part of the federal assignment under contract N 8.4048.2017/4.6 from 31.05.2017.

REFERENCES

[1] SpaceWire Standard. ECSS – Space Engineering. "SpaceWire – Links, Nodes, Routers and Networks". ECSS-E-ST-50-12C, July 2008.

[2] Olenev V.L., Lavrovskaya I.I. "Computer-aided design system for on-board SpaceWire networks simulation and design", Proceedings of SUAI Scientific session, Part 1, Technical sciences/ Saint Petersburg, SUAI, 2017.

[3] ESA. Standard ECSS-E-ST-50-52C, SpaceWire — Remote memory access protocol. Noordwijk : Publications Division ESTEC, February 5, 2010.

[4] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov "*STP-ISS Transport Protocol for Spacecraft On-board Networks*", Proceedings of 6th International SpaceWire Conference 2014 Program; Greece, Athens, 2014. pp. 26-31.

[5] ESA (European Space Agency). Standard ECSS-S-ST-50-53C, "Space engineering. SpaceWire — CCSDS Packet Transfer Protocol. European cooperation for space standardization". Noordwijk: ESA Publications Division ESTEC, 2010. 21 p.

[6] Sandia National Laboratories, Sandia report SAND2011-3500, "Joint Architecture Standard Reliable Data Delivery Protocol Specification". Sandia National Laboratories, Albuquerque, New Mexico, 2011. 39 p.

[7] EADS Astrium (European Aeronautic Defence and Space Company), SMCS-ASTD-PS-001 1.1, "STUP SpaceWire Protocol Protocol Specification", 2009. 7 p.

[8] Sheynin Y., Suvorova E., Schutenko F., Goussev V. "Streaming Transport Protocols for SpaceWire Networks", International SpaceWire Conference 2010, Saint Petersburg, 2010.

[9] Syschikov, A., Sheynin, Y., Sedov, B., Ivanova, V. "*Domain-specific programming environment for heterogeneous multicore embedded systems*", International Journal of Embedded and Real-Time Communication Systems, Volume 5, Issue 4. 2014, pp. 1-23.

[10] Elena Suvorova, Yuriy Sheynin, Nadezhda Matveeva, "Reconfigurable NoC development with fault mitigation", *in Proceedings of the 18th FRUCT & ISPIT Conference*, Technopark of ITMO University, Saint-Petersburg, Russia, 2016, pp. 335-344.

[11] Rui Borralho, Pedro Fontes, Ana Antunes, Fernando Morgado Dias, "A Matlab Tool for Analyzing and Improving Fault Tolerance of Artificial Neural Networks", *in IFAC Proceedings Volumes*, Volume 42, Issue 19, 2009, Pages 158-163.

[12] Alexey Syschikov, Elena Suvorova, Yuriy Sheynin, Boris Sedov, Nadezhda Matveeva, Dmitry Raszhivin, "Toolset for SpaceWire Networks Design and Configuration", *in Proceedings of the 5th International SpaceWire Conference 2013*, 2013, pp.149-153.

[13] Seiculescu C. et al. A method to remove deadlocks in networks-on-chips with wormhole flow control // *Proceedings of the Conference on Design, Automation and Test in Europe.* – European Design and Automation Association, 2010. – C. 1625-1628.

[14] Duato J., Yalamanchili S., Ni L. M. *Interconnection networks: an engineering approach.* – Morgan Kaufmann, 2003.\

[15] D.V. Buzdalov et al, Integrated modular avionics system design tools // Proceedings of Institute for System Programming of the RAS. – 2014. – V. 26. – №. 1.

[16] TTEPlan, TTEthernet/AFDX® network planning tool. Web: https://www.tttech.com/products/aerospace/development-test-vv/development-tools/tte-plan/

[17] Aircraft Data Network, Part 1: *Systems Concepts and Overview*, 2002.

[18] Aircraft Data Network, Part 7: Deterministic Networks, 2003.

[19] Hou Jianru, Chen Xiaomin, Sun Huixian, *"An OPNET Model of SpaceWire and Validation"*, Proceedings of the 2012 International Conference on Electronics, Communications and Control, Zhoushan, 2012. pp. 792-795

[20] B. Dellandrea, B. Gouin, S. Parkes, D. Jameux, *"MOST: Modeling of SpaceWire & SpaceFiber Traffic-Applications and Operations: On-Board Segment"*, Proceedings of the DASIA 2014 conference, Warsaw, 2014.

[21] Thales Alenia Space, *"Modeling Of SpaceWire Traffic"*, Project Executive Summary & Final Report, 2011, 25 p.

[22] NS-3 Manual, "NS-3 Network Simulator", 2017, 165 p.

[23] B. van Leeuwen, J. Eldridge, J. Leemaster, *"SpaceWire Model Development Technology for Satellite Architecture"*, Sandia Report, Sandia National Laboratories 2011, 30 p.

[24] Mirabilis Design, *"Mirabilis VisualSim data sheet"*, 2003. 4 p.

[25] Lavrovskaya I., Olenev V., Korobkov I. "Fault-Tolerance Analysis Algorithm for SpaceWire Onboard Networks" *in Proceedings of the 21st Conference of Open Innovations Association FRUCT,* University of Helsinki, Helsinki, Finland, 2017, pp. 217-223.

[26] Lavrovskaya, I., & Olenev, V. (2018, May). Network Topology Transformation for Fault Tolerance in SpaceWire Onboard Networks. In *2018 22nd Conference of Open Innovations Association (FRUCT)* (pp. 131-137). IEEE.

[27] Kurbanov, L., Rozhdestvenskaya, K., Suvorova, E. (2018, May). Deadlock-Free Routing in SpaceWire Onboard Network. In *2018 22nd Conference of Open Innovations Association (FRUCT)* (pp. 107-114). IEEE.